Le multimédia Images et audiovisuels

Stéfane Paris

Le multimédia

A Tenzing

LAVOISIER
11, rue Lavoisier
75008 Paris

www.hermes-science.com
www.lavoisier.fr

ISBN 978-2-7462-2350-9
ISSN 1242-7691

Printed and bound in England by Antony Rowe Ltd, Chippenham, September 2009.

Le multimédia

images et audiovisuels

Stéphane Paris

Hermes Science publications

Lavoisier

Collection dirigée par Jean-Charles Pomerol

CHRISMENT Claude *et al.* – *Bases de données relationnelles*, 2008.

BANÂTRE Michel *et al.* – *Informatique diffuse*, 2007.

BARTHELEMY Pierre, ROLLAND Robert et VERON Pascal – *Cryptographie*, 2005.

CARDON Alain – *La complexité organisée : systèmes adaptatifs*, 2004.

FOURNIER Jean-Claude – *Théorie des graphes et applications*, 2005.

PARIS Stéphane – *Le multimédia et la compression*, 2009.

PIERSON Jacky – *La biométrie*, 2007.

POLI Alain et GUILLOT Philippe – *Algèbre et protection de l'information*, 2005.

VARRETTE Sébastien et BERNARD Nicolas – *Programmation avancée en C avec exercices corrigés*, 2006.

VERDRET Philippe – *De Perl à Java : programmation des expressions régulières*, 2004.

Table des matières

Préface

En tant que maître de conférence en informatique, passionné par l'image, j'ai suivi la naissance du multimédia avec intérêt. Et, l'idée d'écrire ce livre s'est imposée assez naturellement : la formation proposée aux étudiants en informatique doit comporter les fondements et les outils du multimédia.

Le statut du multimédia est similaire à celui de l'architecture des ordinateurs. Mise à part quelques exceptions, les informaticiens ne créent pas de nouveaux micoprocesseurs, ne câblent pas de circuits intégrés. Cependant, ils doivent connaître le fonctionnement des machines (présentes et futures) sur lesquelles ils vont programmer.

De même, le multimédia doit faire partie de la culture scientifique informatique. L'informaticien ne peut programmer et gérer le *web* de demain sans connaître les outils qu'il manipule. Ces outils ne sont autres que les audiovisuels, les images, les hypertextes, les graphiques, etc. En un mot : le multimédia.

Cette idée s'est concrétisée quand j'ai proposé un cours de multimédia. Initialement, j'avais envisagé de ne faire que quelques heures. Je voulais proposer aux étudiants un survole du sujet. Mais très vite, ce volume horaire est apparu trop restreint pour répondre à toutes leurs questions. J'ai donc déposé sur mon site Internet des références en adéquation avec le public informatique. Je n'ai, hélas, trouvé qu'un seul livre qui aborde les fondements du multimédia de manière compréhensible – sans prérequis – pour des étudiants informaticiens : *Fundamentals of multimedia*, par Z.N. Li et M.S. Drew, aux éditions Prentice-Hall.

Pour conclure cette préface, je tiens à remercier Guy Bourhis, Francois Brucker, Azzeddine Kaced et Georges Paris pour leurs relectures et leurs critiques ; Mark S. Drew, Ze-Nian Li, Dale Purves, David Salomon et Jacques Weiss pour m'avoir généreusement offert de leurs temps et de leurs matériels ; l'Université Paul Verlaine de

Metz de m'avoir offert un congé d'enseignement qui a facilité la rédaction de ce livre ; les créateurs du logiciel TeXmacs.

Enfin, je remercie tout particulièrement Elisabeth, Evelyne et Sylvie pour leur aide et leur soutien dans ce projet.

Avant-propos

Les deux volumes de ce livre présentent en trois parties l'ensemble des outils actuels (en 2008) du multimédia. Le premier volume se veut une *connaissance à long terme* décrivant les théories du multimédia. Le deuxième volume est plus appliqué et, donc, plus sujet à se *démoder* avec l'évolution de la technologie et de l'économie de marché. Mais, les modes vont et viennent avec des modifications, des variantes qui ne sont pas toujours radicales. Par exemple, le passage de MPEG2 à MPEG4-AVC est fait d'une multitude d'optimisations qui rend les clips vidéo accessibles depuis un téléphone mobile. Cependant, les concepts et les algorithmes utilisés par MPEG4-AVC restent, dans leurs lignes générales, identiques à ceux de MPEG2.

Plus précisément, le premier volume traite des *théories de la compression*. Il aborde en trois volets les principaux concepts nécessaires à la compréhension de l'ensemble des outils du multimédia décrit dans les deux autres parties.

Le premier volet (chapitre 2, volume 1) est le fondement qui a donné naissance à tous les outils qui suivent. Il survole l'ensemble des *transformées de fonctions*. Ces transformées amènent à considérer une fonction non seulement dans le domaine *temporel* ou *spatial*, mais aussi, dans le domaine *fréquentiel*.

Récemment découvert, un ensemble de transformées, issu de la convergence entre la théorie des signaux et l'analyse mathématique, permet d'observer une fonction dans le domaine *spatiofréquentiel*.

Bien que ces notions de transformées reposent sur des théories mathématiques avancées, nombre de raccourcis sont pris pour ne pas trop s'éloigner de l'objectif initial de ce livre; à savoir, donner une *description intuitive* des outils du multimédia.

Ce volet a été conçu pour apporter le minimum nécessaire à toute personne soucieuse d'approfondir un point de vue théorique précis. Des références sont insérées régulièrement comme des clés pour tel ou tel aspect des tranformées. De plus, des annexes viennent compléter certains points.

Le deuxième volet (chapitre 3, volume 1) étudie tout ce qui concerne (1) la numérisation et (2) la compression des signaux.

La *numérisation*, ou, par anglicisme, la *digitalisation*, est la première brique de l'édifice multimédia. Elle définit les propriétés qu'un *convertisseur* (par exemple, un scanner familial) doit avoir pour transformer un *signal analogique* en un *signal numérique* ou celles qu'un *capteur numérique* (par exemple, un appareil photographique numérique) doit avoir pour faire une capture numérique correcte.

Ces propriétés – et les théorèmes qui en découlent – ont été vus dans le premier volet de cette partie. L'analyse fréquentielle du signal, quelle que soit sa nature, est primordiale puisque le signal numérique n'est plus décrit par des valeurs continues mais par des valeurs discrètes.

Ainsi, un signal sonore qui est continu dans le temps, est décrit par des valeurs régulièrement espacées dans le temps, quand il est numérisé. De même, une image numérique est enregistrée sur une matrice composée d'éléments unitaires appelés *pixels* (raccourci pour *picture elements*).

La deuxième étude de ce volet est la *compression*. Celle-ci vise à diminuer la place prise par un signal (sur un support mémoire ou sur un réseau). Elle peut se faire en éliminant toute information redondante. Dans ce cas, il n'y a pas de différence entre le signal original et le signal reconstruit après décompression. La compression est alors *sans perte*.

Les deux processus de compression, que sont (a) la quantification et (b) la codification, font référence à la *théorie de l'information*, introduite par Claude Shannon. Elle est donc brièvement présentée. Ses liens avec la quantification et la codification sont mis en exergue :

1) la *quantification* est, à la fois, un élément de la numérisation et un élément de la compression :

a) elle est un élément de la numérisation car les amplitudes des signaux continus sont réelles. Et, chacun sait que la représentation des réels sur une machine n'est qu'une approximation des réels mathématiques. Dans certains cas, on peut vouloir des signaux numériques à amplitudes entières. Par exemple, les images au format RGB ont des valeurs de pixels entières, comprises entre 0 et 255. La quantification s'apparente alors à un découpage de l'axe des réels mathématiques en intervalles, tel que chaque

intervalle ait un représentant enregistré sur la machine. Ainsi, à chaque amplitude continue correspond un intervalle. L'amplitude est alors remplacée par le représentant de cet intervalle,

b) la quantification est également un élément de la compression car elle intervient souvent après la numérisation pour augmenter le taux de compression (c'est-à-dire, pour diminuer la taille du signal compressé). Les pertes sont alors importantes et ne doivent pas avoir lieu n'importe où, n'importe quand ;

2) la *codification* récupère le résultat provenant de la quantification du signal numérique. Elle transforme ce signal en un *flux binaire* pour être stocké sur un support adéquat (CD-ROM, DVD, etc.) ou pour être transmis sur un réseau.

Mais, une *compression avec pertes* peut s'avérer obligatoire si la taille engendrée par la compression sans perte n'est pas suffisament faible. On a vu que la quantification est souvent réutilisée pour répondre à ce besoin. Mais, il faut alors contrôler les pertes engendrées.

Dans le troisième et dernier volet de ce premier volume (chapitre 4, volume 1), l'utilisateur est remis au centre du processus afin de contrôler les pertes inévitables si l'on veut transmettre ou stocker un signal.

Ainsi, deux questions fondamentales se posent. Qu'est-ce que l'ouïe ? Qu'est-ce que la vue ? Questions qui ont permis de construire des algorithmes de compression engendrant des pertes qui ne sont pas (trop) génantes pour l'utilisateur.

Ce troisième volet présente, donc, succinctement, les modèles de compression choisis pour les signaux sonores et visuels. Ces modèles autorisent des pertes d'information quand la perception humaine est peu sensible tout en conservant une relativement grande précision quand la perception humaine est plus sensible.

Le deuxième volume présente en deux parties les formats images et les formats audiovisuels du multimédia. La première partie se divise en trois chapitres qui définissent trois modèles de représentation des images.

Le premier chapitre (chapitre 2, volume 2) est dédié aux modèles de compression sans perte des images. Les codeurs décrits servent de référence aux outils de compression avec pertes des images et des audiovisuels.

Le deuxième chapitre (chaitre 3, volume 2) présente le standard grand public actuel (en 2008); à savoir, le format JPEG. Il est d'autant plus incontournable que les premières versions du format MPEG (MPEG1 et MPEG2) le prennent comme base pour la compression vidéo.

Enfin, le troisième et dernier chapitre de cette première partie (chapitre 4, volume 2) décrit le standard JPEG2000. Bien que mis à l'index pour des raisons économiques, il utilise, pour une des premières fois dans le cadre de la compression d'images, nombre d'outils très performants. D'ailleurs, ces outils sont utilisés par la partie vidéo du standard MPEG4.

La deuxième partie de ce volume présente la compression audiovisuelle à travers l'évolution du format MPEG.

Le standard MPEG1 est décrit en premier (chapitre 5, volume 2). Il pose les bases de la compression audio et vidéo.

Puis, le chapitre suivant (chapitre 6, volume 2) présente le standard MPEG2. Celui-ci améliore certains aspects de la compression audio et vidéo proposés par le standard MPEG1. Mais, surtout, il introduit de nouveaux concepts, comme les *profils*, les *niveaux* et la *gestion multimédia* (DSM-CC). Ces concepts permettent une adaptation du format MPEG aux exigences des réseaux. De plus, il autorise une interaction (encore faible) entre l'utilisateur et le produit multimédia.

Le dernier chapitre (chapitre 7, volume 2) aborde le standard MPEG4 et sa version développée (en 2008) MPEG4-AVC.

La compression est révolutionnée par la *modélisation objet* de tout élément audio, visuel, texte, graphique, etc. Ceci reste encore fortement théorique car cette modélisation requiert des outils qui sont encore (en 2008) au stade de la recherche.

Contrairement à ce que son nom laisse entendre, MPEG4-AVC est plus une version optimisée du standard MPEG2 qu'une version du standard MPEG4.

Toutefois, le modèle du standard MPEG4 peut déjà être utilisé dans des situations suffisamment contraintes. Dans ces cas de figure, la modélisation objet permet une forte interactivité où l'utilisateur final peut intervenir au niveau de la présentation du produit multimédia (lors du décodage), mais aussi, au niveau de la conception du produit multimédia (lors du codage). Elle permet également de fusionner en un même produit multimédia des *données synthétiques* (textes, graphiques, sons et images de synthèse, etc.) et des *données naturelles* (photographies, sons et audiovisuels enregistrés, etc.).

Il s'agit là de l'avenir du multimédia. Quels ques soient les outils qui seront développés, l'interaction avec des données naturelles et synthétiques sera de plus en plus forte.

Comment lire ce livre ?

Il est évident que l'on peut lire ce livre dans sa globalité en commençant par le premier volume. Toutefois, j'ai tenté de rendre les volumes et les chapitres les plus indépendants possibles les uns des autres. Quand cela est nécessaire, j'ai inséré des références à des sections de chapitres précédents afin de permettre au lecteur d'approfondir certains aspects.

Par exemple, le chapitre traitant du standard MPEG1 peut être lu séparément des autres. Mais, puisqu'il réutilise les techniques du standard JPEG, celles-ci sont brièvement décrites dans le chapitre MPEG1. Des références au chapitre JPEG du volume 2 permettent d'approfondir la lecture. De même, le chapitre JPEG fait référence au chapitre 2 du volume 1 pour une description plus détaillée de la transformée en cosinus discrète.

Sur le site http://sites.google.com/site/intromm/ vous trouverez les images en couleurs de ce livre ainsi que des compléments d'information.

Chapitre 1

Introduction

Ce deuxième volume présente en deux parties les standards dédiés aux images numériques et les standards dédiés aux audioviduelles numériques.

1.1. Les images numériques

Les images numériques sont compressées afin de permettre leurs stockages et/ou leurs transferts *via* un réseau. En conservant les images au format brut que délivre les capteurs numériques, ces tâches ne sont pas réalisables. Il faut utiliser des outils de compression adaptés aux images. Ces outils opèrent soit en restant fidèles à la qualité des images soit en autorisant des déformations.

Le premier chapitre de cette partie (chapitre 2) est dédié aux techniques de *compression sans perte*. Une image peut alors être compressée et décompressée autant de fois que souhaiter, elle restera identique à l'image originale.

Le chapitre 3 présente le, bien connu, standard JPEG de compression avec perte. Contrairement aux techniques sans perte, plus on répète le cycle compression/décompression plus l'image se détériore. L'intérêt d'une compression avec perte réside dans le gain de place gagné par rapport aux techniques sans perte. Ceci signifie qu'une image est compressée peu de fois (une seule fois en général) mais qu'elle est décompressée autant de fois que voulu. La compression étant effectuée moins fréquemment que la décompression, le processus de compression peut être sophistiqué et relativement lent. Le processus de décompression doit être relativement simple et efficace.

Le chapitre 4 présente le standard JPEG2000. Celui-ci améliore nettement les taux de compression proposés par le standard JPEG tout en conservant une qualité d'image identique voire meilleure.

1.2. Les audiovisuels numériques

Cette deuxième partie décrit les différents formats MPEG dédiés à la compression et à la présentation d'audiovisuels. Il s'agit des standards MPEG1, MPEG2 et MPEG4.

Le chapitre 5 décrit le standard de compression audiovisuelle MPEG1.

Le chapitre 6 présente le standard MPEG2 qui améliore les taux des compression du standard MPEG1 et introduit les premières interactions entre l'utilisateur et le support multimédia.

Le chapitre 7 décrit le standard MPEG4 qui propose une *révolution* dans le domaine de l'audiovisuel. L'interaction est le maître mot. L'utilisateur peut interagir tout au long de la durée de vie du support multimédia; de la création à la présentation. Ce standard propose également de fusionner les données synthétiques et les données naturelles.

PREMIÈRE PARTIE

Compression et représentation d'images

Synopsis

Faire une étude exhaustive des techniques de compression d'images aurait rendu confuses les explications tant la liste en est longue et qu'elles sont parfois très proches les unes des autres. Aussi, cette partie présente ces techniques en trois catégories.

Chaque catégorie est décrite par un chapitre.

Le premier chapitre (chapitre 2) traite de la *compression sans perte*. Ce tour d'horizon commence avec le format GIF qui est le précurseur des compressions sans perte. Il repose sur l'utilisation du codeur à dictionnaire LZ77 qui est un outil de base dans tous les systèmes d'exploitation. Son successeur, le format PNG, est ensuite présenté. Il étend les capacités du format GIF aux images en « vraies couleurs » (c'est-à-dire codant les pixels sur 8 bits au minimum) ; la partie LossLess du standard JPEG et le standard JPEG-LS sont enfin décrits. Du format GIF au format JPEG-LS, les taux de compression vont en augmentant. Ce gain se fait au prix d'une complexification algorithmique et/ou calculatoire.

Le chapitre 3 détaille le standard JPEG qui a fortement contribué à l'entrée du numérique dans notre quotidien. Il permet de régler le taux de compression, c'est-à-dire de gérer les pertes engendrées : *plus le taux de compression est fort, plus les pertes sont importantes, donc, visibles.*

Ce standard utilise une DCT afin de tirer profit d'une analyse fréquentielle de l'image. Cette analyse offre la possibilité de sélectionner les éléments supprimés lors de la compression. Cette sélection est faite de manière à géner le moins possible les utilisateurs. Différents modes de transmission et d'affichage de l'image décompressée sont permis afin de s'adapter aux contraintes dues aux réseaux et à l'utilisateur.

Avec le standard JPEG2000, le dernier chapitre de cette partie (chapitre 4) décrit un nouveau type de compression. Le système de compression n'est plus dédié aux seules

images naturelles (c'est-à-dire les photographies). Il tend vers une intégration multimédia des images. Une image peut aussi bien être une photographie familiale qu'une image haute résolution, délivrée par un satellite, par exemple. Cette image peut être accompagnée d'informations annexes : logo, textes, images synthétiques (graphiques), etc. Les différentes composantes des images sont alors codées indépendamment les unes des autres. L'analyse fréquentielle est localisée grâce à l'utilisation d'une transformée en ondelettes. Cette approche spatiale permet un contrôle plus fin de la compression; les hautes fréquences sont localisées et la compression peut être plus forte sur ces zones. Une étude du comportement entropique du codage a permis de définir un ensemble d'outils augmentant sensiblement les taux de compression. Il est à noter que le standard autorise, avec les mêmes outils, une compression avec ou sans perte d'images comportant de multiples et diverses composantes.

Au vu de leurs performances, les outils développés par les standards JPEG et JPEG2000 – principalement les analyses fréquentielles, les quantificateurs et les codeurs entropiques – sont réutilisés par les standards vidéo MPEG, décrits dans la dernière partie de ce livre (partie II).

Chapitre 2

GIF, PNG, LossLess-JPEG et JPEG-LS

Il est impossible d'être exhaustif dans la présentation des standards images de compression sans perte. Ils sont trop nombreux pour cela.

Cependant, on peut les regrouper en catégories suivant les algorithmes de codage utilisés et suivant les objectifs visés. Ce regroupement, somme toute assez subjectif, est décrit par les représentants suivants :

1) le standard GIF, étudié en section 2.1, est l'*ancêtre* des formats d'échanges d'images ;

2) le standard PNG, étudié en section 2.2, est son successeur désigné ;

3) le standard LossLess-JPEG, étudié en section 2.3, est la version sans perte du format JPEG ;

4) le standard JPEG-LS, étudié en section 2.4, représente les nouveaux standards de compression sans perte, visant des applications allant du fax à Internet, qui offrent des algorithmes efficaces et légers pour des taux de compression supérieurs à ses prédécesseurs.

La partie LossLess-JPEG est présentée indépendamment du standard JPEG, car il s'agit de deux techniques distinctes. Entre autres différences, la transformée DCT utilisée par JPEG ne l'est pas par LossLess-JPEG. Le standard JPEG est, en fait, plus connu dans sa version compression avec perte, qui est détaillée au chapitre 3 suivant.

Le standard JPEG-LS n'a rien de commun avec le standard JPEG, sinon son nom. Ses taux de compression le place en tête des algorithmes de compression sans perte présentés dans ce chapitre. De plus, son coût algorithme est identique, voire plus faible que ceux des autres standards.

2.1. GIF : *Graphic Interchange Format*

C'est l'un des plus anciens *formats d'échange d'images* entre ordinateurs [COM 87]. Il a été immédiatement intégré comme format de transfert d'images par l'ensemble des navigateurs Internet, tout comme l'ont été, ensuite, les formats PNG et JPEG.

Le cœur du standard est une version fenêtrée de l'algorithme sans perte LZW. C'est donc un codeur à dictionnaire dynamique (voir chapitre 3 volume 1) initialement utilisé par :

– les outils systèmes d'archivage tels que *compress* d'*Unix*, *rar*, *zip* ou encore *gzip*

– la recommandation V42bis de l'UIT pour la communication de données sur le réseau téléphonique.

Dans ces circonstances, c'est tout naturellement que l'un des premiers formats d'échange d'images a opté pour ce type de codage sans perte. Mais, une des limites fondamentales de ce codeur est sa technique de lecture séquentielle qui ne tient nullement compte du contexte.

Comme on l'a vu dans le chapitre 3 du volume 1, toute source dont les échantillons sont corrélés ne peut-être codée efficacement qu'en intégrant le contexte de chaque échantillon. Il est évident que les pixels d'une image ne sont pas sans relation les uns avec les autres.

GIF est donc idéal pour des *images synthétiques*, tels les logos, qui présentent souvent des régions uniformes. En revanche, pour des *images naturelles*, c'est-à-dire photographiques, la recherche de séquences répétitives est plus hasardeuse. Les formats PNG et surtout JPEG sont alors bien plus adaptés.

Le standard GIF utilise la structure matricielle représentant les images : structure souvent appelée, par anglicisme, *bitmap* ou *raster*. Le nombre de lignes et de colonnes de la matrice correspond, respectivement, à la *hauteur* et à la *largeur* de l'image[1].

Une *table de références des couleurs* (*color lookup table*) est associée à cette matrice. Chaque pixel est décrit par un coefficient dans la matrice. Ce coefficient est une entrée dans la table de références. La couleur (un triplet RGB) correspondant à cette entrée est celle du pixel.

La carte des pixels a une profondeur pouvant aller de 2 à 8 bits. La profondeur la plus faible permet de coder les images binaires (images à deux niveaux de gris : le

1. Le pixel origine est en première ligne, première colonne de la matrice.

blanc et le noir) alors qu'une profondeur de 8 bits assure le codage d'images à 256 couleurs ou à 256 niveaux de gris[2].

La structure d'une image au format GIF est illustrée par le schéma de gauche de la figure 2.1. La *signature* permet de reconnaître le format. Elles est constituée des six octets : `G I F 8 7 a`. Le nombre 87 est là pour rappeler que le standard est apparu en 1987. Le *descripteur écran* informe sur les caractéristiques de l'écran virtuel. Il est utilisé, lors de l'affichage réel, *via* une transformation linéaire avec les caractéristiques de l'écran physique. Ses valeurs sont données dans le schéma de droite de la figure 2.1. La largeur et la hauteur sont codées sur deux octets chacune. Le bit `M` est à `1` si une table de références globale des couleurs (*color lookup table*) est utilisée ; $(c_r + 1)$ est le nombre de bits pour la résolution en couleur; $(n + 1)$ est le nombre de bits par pixel avec n qui varie de 0 à 7[3]. Le champ *arrière-plan* est l'indice de la couleur de fond dans la table de références globale si celle-ci existe, sinon, dans la table par défaut.

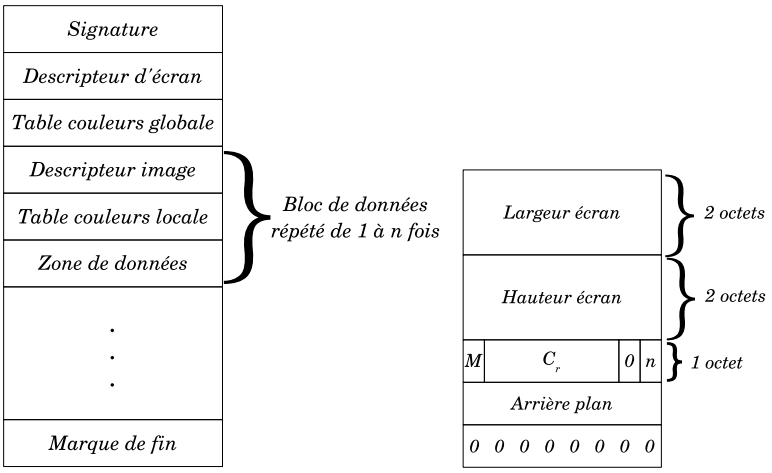


Figure 2.1. *A gauche : la structure du format GIF ; à droite : les octects du descripteur d'écran*

Si `M=1`, la table de références globale des couleurs a 2^{n+1} entrées. Chaque entrée est codée sur trois octets pour les valeurs de rouge, vert et bleu respectivement. Lors du codage, la couleur de chaque pixel est remplacée par l'entrée qui lui est la plus proche.

2. Images malhabilement nommées noir et blanc en photographie.

3. Il définit également le nombre maximum de couleurs.

Après cette table globale, vient une suite de triplets <*descripteur image, table des couleurs locale, zone de données*> qui se termine avec la marque de fin «;» (codé sur un octet). Chaque triplet décrit une image.

Le *descripteur image* est donné en figure 2.2. Les positions d'origine haute et gauche, la hauteur et la largeur de l'image sont codées sur deux octets chacune. Le bit `M` prend la valeur :

– `0` : pour indiquer que la table de références globale est à utiliser ;

– `1` : pour indiquer qu'une table locale est définie avec 2^{p+1} entrées.

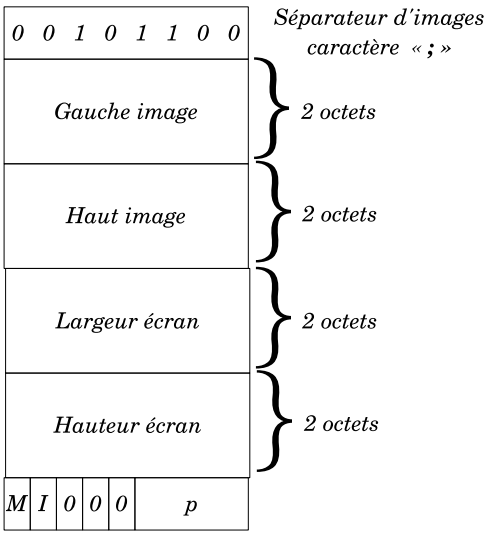


Figure 2.2. *Le descripteur image*

Le bit `I` prend la valeur `0` pour spécifier un codage – et, de surcroît, un affichage – séquentiel. Sinon, il prend la valeur `1` pour stipuler qu'il s'agit d'un codage *entrelacé*. En mode séquentiel ($I = 0$), la *zone de données* du triplet est la *bitmap* de l'image. Chaque pixel a pour valeur une entrée dans la table de références associée (locale ou globale).

En mode entrelacé ($I = 1$), les pixels sont rangés en quatre passes successives. La première passe commence à la première ligne et récupère toutes les lignes avec un pas de huit. Ensuite, la deuxième passe conserve le pas de huit, mais, commence à la cinquième ligne. La troisième passe commence à la troisième ligne avec un pas de quatre. Enfin, la quatrième passe récupère toutes les lignes restantes ; c'est-à-dire celles commençant à la deuxième ligne et espacées d'une ligne. Le mode entrelacé

permet d'avoir un affichage progressif de l'image. Ceci limite les temps de visualisation pour des débits faibles.

Le champ p n'est utilisé que si le bit `M` est à `1`; autrement dit, quand une table de références locale est définie. A la fin de l'image, la table globale est restaurée.

Un ensemble d'extensions constitué de codes de fonctions peut précéder un descripteur image ou être placé juste avant la marque de fin. Ces extensions et le traitement spécifique de la marque de fin introduisent de l'*intercation* entre l'image GIF et l'application au sein de laquelle elle évolue. Par exemple, lors de l'affichage d'une image au format GIF, le navigateur peut attendre la frappe de la touche entrée ou le clic de la souris avant de continuer l'affichage du reste de la page Internet.

Le fait d'autoriser le stockage de plusieurs images permet de définir des animations. Ces animations restent simples, puisque dès qu'une image est décodée, celle-ci vient remplacer la précédente dans la zone d'affichage.

2.1.1. *Version GIF89a*

Une deuxième version, GIF89a, vient améliorer certains aspects de la version GIF87a [COM 89]. Plusieurs *méthodes d'agencement* (*disposal method*) sont possibles après affichage d'une image. Soit la couleur de fond (champ *arrière-plan* de la figure 2.1) peut être rétablie, soit l'image peut être conservée, soit encore le décodeur doit restaurer ce qu'il y avait avant l'affichage de l'image. Une méthode permet à l'utilisateur de personnaliser l'agencement. Sinon, comme avec la version précédente, il est possible de ne rien prévoir.

La version GIF89a autorise également l'utilisation d'une couleur de transparence pour chaque image. Cette transparence est totale; il n'y a pas de possibilité de régler le niveau de transparence. Un des indices de la table de références, utilisée par l'image concernée, indique la couleur de transparence. Ainsi, lorsque le décodeur rencontre un pixel de cette couleur, il ne change pas la valeur d'affichage pour celui-ci.

Il est aussi possible d'indiquer des délais, entre les affichages des images, afin de personnaliser l'animation.

Enfin, des blocs d'extensions ont été ajoutés pour :

– l'affichage de textes, au même titre que l'affichage des images ;

– l'ajout de commentaires qui peuvent être ignorés par le décodeur ;

– l'association d'applications au fichier GIF.

2.1.2. *Synthèse du format GIF*

Les spécifications liées à la programmation des codeurs et décodeurs des formats GIF se trouvent accessibles sur le site : ftp://ftp.ncsa.uiuc.edu/misc/file.formats/graphics.formats.

Ce format est utilisé pour les *imagettes* qui agrémentent les sites Internet. Ces imagettes sont fixes ou correspondent à de petites animations. La figure 2.3 illustre le gain mémoire offert par le format GIF.

Figure 2.3. *(a) image source non compressée – format BMP – (taille 2,3 Mo); (b) l'image compressée au format GIF (taille 357,4 Ko). On ne constate visuellement (et objectivement) aucune différence malgré la différence en taille de mémoire occupée*

Cependant, les taux de compression (sans perte) ne sont pas suffisant dès lors que les images sont de tailles supérieures à celles des imagettes. Aussi le format PNG a-t-il été introduit pour pallier à ce défaut.

2.2. PNG : *Portable Network Graphics*

Le format PNG opère une compression par *déflation/inflation* (*deflate/inflate*). Il est constitué d'un ensemble de *segments* (*chunks*) :

– une signature sur 8 bits : 137 80 78 71 13 10 26 10 ;

– des segments obligatoires (*critical*) :

- un segment d'entête IHDR, définit les dimensions de l'image, le nombre de bits par plan binaire (b_p), l'espace couleur (*color type*), les paramètres de la prédiction (appelée filtrage dans le rapport [PNG 03]) et le mode d'entrelacement,
- un segment de la palette IPLTE, si *color type* est 3,
- un segment de données IDAT,
- un segment IEND, contenant la marque de fin ;

– des segments annexes :

- publics : définis et ordonnés par le standard,
- privés.

Un segment est une structure qui comprend une taille, un nom, un champ de données et 32 bits CRC (type Ethernet).

2.2.1. *Les espaces de couleurs*

L'image peut être :

1) *color type* 0 ou 4 à niveaux de gris avec ou sans transparence respectivement :

2) *color type* 2 ou 6 en vraies couleurs avec ou sans transparence respectivement :

3) *color type* 3 indexée par une table de références.

Le nombre de bits par plan varie de 1 à 16. Le tableau 2.1 donne les caractéristiques de chaque type de couleurs.

type de couleurs	code	nombre de bits par plan	interprétation
niveaux de gris	0	$b_p = 1, 2, 4, 8, 16$	chaque pixel est une valeur de gris
vraies couleurs	2	$b_p = 816$	chaque pixel est un triplet (R,G,B)
indéxée	3	$b_p = 1, 2, 48$	chaque pixel est un indice dans une table de références, rangée dans le segment ILPTE
niveaux de gris avec transparence	4	$b_p = 8, 16$	chaque pixel est un couple de valeurs (gris, α)
vraies couleurs avec transparence	6	$b_p = 8, 16$	chaque pixel est un quadruplet de valeurs (R,G,B,α)

Tableau 2.1. *Les types de couleurs : α est la valeur d'opacité*

Le canal α d'opacité varie de totalement transparent (0) à totalement opaque $(2^{\mathrm{bp}-1})$. Pour une image à niveaux de gris, les valeurs des pixels sont codées sur deux canaux; pour une image en vraies couleurs, sur quatre canaux :

$$\alpha \in [0, 2^{b_p-1}]$$

Pour les images indéxées, un segment de transparence tRNS contient une table de références. Celle-ci indique les valeurs de transparence des couleurs rangées dans la table de références du segment IPLTE. Les valeurs sont comprises entre 0 et 255 (un octet).

Ainsi, la couleur affichée C, est une combinaison de la couleur de fond B, et de la valeur P, du pixel dans l'image :

$$C = (1 - 2^{-b_p-1}\alpha)P + 2^{-b_p-1}\alpha B$$

Dans le cas d'une image à niveaux de gris, chaque entrée référence une valeur unique. Pour une image en vraies couleurs, chaque entrée référence un triplet de valeurs.

Le segment, tRNS, peut également servir à définir une couleur totalement transparente pour les images à niveaux de gris ou en vraies couleurs. Dans ce cas, les valeurs des autres indices sont forcément totalement opaques. On retrouve alors la définition de la transparence du format GIF.

2.2.2. *Les modes d'entrelacement*

Il existe deux modes d'affichage : entrelacé ou non. Sans entrelacement, les pixels sont rangés ligne par ligne, en commençant par le pixel en haut à gauche. Le mode entrelacé peut, d'une certaine manière, s'apparenter à une description multirésolution [BOU 97]. Il fonctionne sur des blocs de taille 8×8.

$L =$ENTRELACS(I)
1 *// initialisation de la liste des blocs à traiter*
2 $B \leftarrow \{$ blocs 8×8 de $I\}$
3 $L \leftarrow \emptyset$
4 **tant que** $B \neq 0$
5 **faire** Extraire bloc b de B
6 $L \leftarrow L \oplus \text{valeur}_{\text{pixelHautGauche}}$
7 **si** TEX$blocdeplusieurspixels$
8 **alors si** bloc de forme carré
9 **alors** Diviser b en deux blocs b_1 et b_2 , suivant les colonnes
10 **sinon** Diviser b en deux blocs, b_1 et b_2 , suivant les lignes
11 $B \leftarrow B \oplus \{b_1, b_2\}$

La liste B contient les blocs à traiter. Cette liste est initialisée avec l'image découpée en blocs de taille 8×8. Chaque bloc est représenté par son pixel haut gauche. La liste L récupère cette valeur. Ensuite le bloc est scindé en deux sous-blocs, b_1 et b_2, de tailles identiques. Ces nouveaux blocs sont rangés en queue de liste B. Sept passes sont ainsi effectuées et les pixels sont rangés dans la liste L suivant l'ordre indiqué par la figure 2.4.

L'algorithme d'affichage opère le traitement inverse à l'aide de la liste L. Connaissant la taille des blocs et les dimensions de l'image, il est aisé de connaître le nombre de blocs dans l'image. Chaque bloc est affiché avec une couleur unie définie par son pixel haut gauche extrait de la liste L.

1	6	4	6	2	6	4	6
7	7	7	7	7	7	7	7
5	6	5	6	5	6	5	6
7	7	7	7	7	7	7	7
3	6	4	6	3	6	4	6
7	7	7	7	7	7	7	7
5	6	5	6	5	6	5	6
7	7	7	7	7	7	7	7

Figure 2.4. *Ordre de rangement des pixels : pour des valeurs identiques, le parcours se fait de haut en bas, ligne par ligne, et de gauche à droite sur chaque ligne*

Lorsque tous les blocs sont traités, ils sont divisés en deux. Ceci a pour effet de doubler la résolution en colonnes. Le procédé est itéré jusqu'à obtenir une taille 1×1 de bloc. A l'affichage, l'image apparaît grossièrement pour être ensuite affinée jusqu'à sa résolution initiale. Le principe est illustré sur un bloc en figure 2.5.

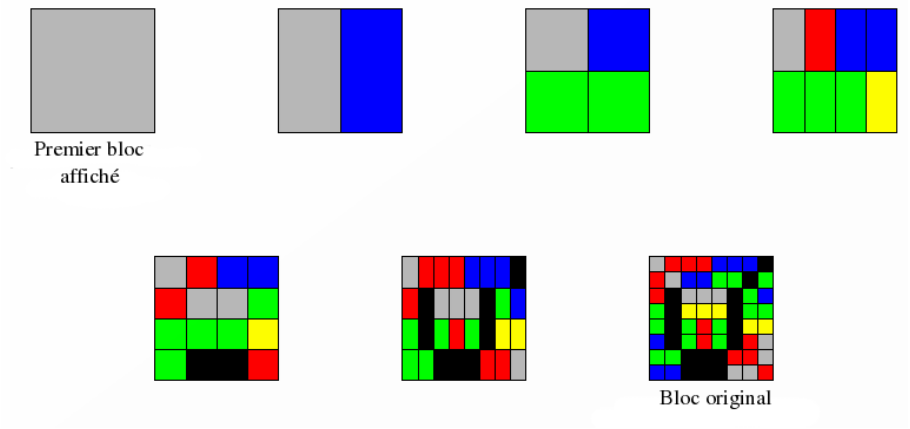


Figure 2.5. *L'affichage entrelacé d'un bloc est progressif*

2.2.3. *L'algorithme de compression*

La compression opère en deux étapes successives sur chacun des canaux. Premièrement, une prédiction adaptative (ADPCM). permet de mettre en forme les données. Puis, la technique de *déflation* effectue la compression proprement dite à l'aide d'un codage préfixé et d'un jeu de tables de Huffman [DEU 96].

2.2.3.1. *La prédiction adaptative*

La prédiction enregistre l'erreur :

$$E_{i,j} = B_{i,j} - \widehat{B}_{i,j}$$

où $B_{i,j}$ est l'*octet* en ligne i et colonne j de l'image et où $\hat{B}_{i,j}$ est la prédiction associée. Il est à noter que la prédiction ne porte pas sur les valeurs des pixels, mais, sur les octets. Ainsi, lorsque le nombre b_p de bits par canaux est de moins de 8 bits, les valeurs des pixels sont regroupées pour former des octets. Mais, en pratique, le standard recommande de ne pas opérer de prédiction (type 0, voir ci-après) pour des profondeurs de plans inférieures à l'octet. Quand $b_p = 16$ bits, la prédiction se fait séparément sur les octets de poids fort et sur les octets de poids faible des valeurs des pixels.

La valeur de prédiction $\hat{B}_{i,j}$, peut être obtenue de cinq manières différentes; d'où l'adjectif adaptatif de la prédiction. Chaque ligne est traitée indépendamment des autres avec la méthode la plus adéquate ; c'est-à-dire celle dont la somme des différences absolues des erreurs est minimale :

$$\sum_{i,j} \mathrm{abs}(E_{i,j})$$

Techniquement parlant, l'ensemble des cinq méthodes est appliqué. Puis, pour chaque ligne la meilleure est choisie.

Toutes les prédictions reposent sur le voisinage direct, montré en figure 2.6, du pixel x, en cours de codage : le pixel a est le prédécesseur de x sur la même ligne, c le prédécesseur sur la ligne précédente et b est situé à la même position que x, mais, sur la ligne précédente également.

c	b
a	x

Figure 2.6. *Le voisinage de prédiction : la figure montre la position des pixels alors que la prédiction porte sur les octets de ces pixels. Il n'y a concordance exacte que lorsque la profondeur des canaux* $b_p = 8$

Les cinq méthodes sont :

1) aucune prédiction : $\hat{B}_{i,j} = 0$; (le choix recommandé quand $b_p < 8$) ;

2) le voisin de gauche : $\hat{B}_{i,j} = a$;

3) le voisin au-dessus : $\hat{B}_{i,j} = b$;

4) la moyenne : $\hat{B}_{i,j} = \left\lfloor \frac{a+b}{2} \right\rfloor$;

5) l'estimation plane, appelée *Paeth*[4]; il s'agit de la différence entre la valeur en x et la valeur fictive p à la même position, mais, appartenant au plan défini par les trois points a, b et c :

$$p = \frac{1}{2}(a+b) + (\frac{1}{2}(a+b) - c) = a + b - c$$

L'algorithme est le suivant :

```
PR = PAETH(a, b, c, x)
 1  p ← a + b − c
 2  pa ← abs(p − a)
 3  pb ← abs(p − b)
 4  pc ← abs(p − c)
 5  si pa ⩽ pb  et  pa ⩽ pc
 6     alors Pr ← a
 7     sinonsi pb ⩽ pc
 8              alors Pr ← b
 9              sinon Pr ← c
```

La figure 2.7 illustre cette situation.

REMARQUE 2.1.– *L'ordre de sélection doit être respecté pour que le codeur et le décodeur soient en accord.*

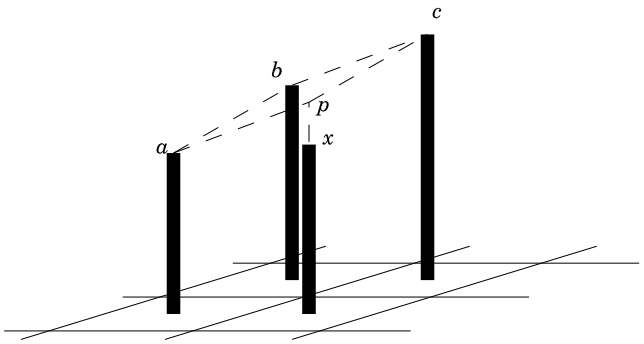


Figure 2.7. *La prédiction Paeth.*

4. En l'honneur de son créateur : Alan W. Paeth.

2.2.3.2. *La déflation/inflation*

La déflation/inflation utilise une fenêtre glissante sur les octets à coder. L'algorithme est une variante du LZ77 qui est lui-même un *parent* du LZW. La fenêtre est composée de deux parties : un tampon de prélecture et une zone de recherche contenant soit les caractères[5] déjà codés, dans le cas du codeur, soit les caractères déjà décodés, dans le cas du décodeur. Cette fenêtre varie entre 256 et 32 768 octets.

L'algorithme de codage identifie la plus grande séquence dans la zone de prélecture déjà présente dans la zone de recherche. Lorsque celle-ci est trouvée, elle est codée sous la forme d'un couple <*distance, longueur*> où *distance* indique le déplacement arrière pour connaître le début de l'occurrence présente dans la zone de recherche et où *longueur* indique la taille de la séquence. Certains caractères du tampon de prélecture ne correspondent à aucune séquence dans la zone de recherche. Ces caractères sont appelés des *lettres* et sont codés sans être transformés.

Ensuite, une paire d'arbres de Huffman est utilisée pour un codage efficace, respectivement, des couples <*distance, longueur*> et des *lettres*.

Les lettres – d'alphabet $\{0 \cdots 255\}$ – et les longueurs – d'alphabet $\{3 \cdots 258\}$ – sont fusionnées en un seul alphabet $\{0 \cdots 285\}$:

1) l'intervalle $0 \cdots 255$ est réservé aux lettres ;

2) la valeur 256 sert de marque de fin de bloc ;

3) les longueurs sont codées dans l'intervalle $257 \cdots 285$, avec une possibilité de bits supplémentaires, comme le montre le tableau 2.2.

Puis, la table de Huffman, donnée par le tableau 2.3, est utilisée pour coder les symboles. Par exemple, les longueurs 13 et 14 ont le même symbole 266 et pour bit supplémentaire `0` et `1` respectivement. La longueur 13 est finalement codée `0001110|0` et la longueur 14, `0001110|1`.

Quant aux distances – d'alphabet $\{1 \cdots 32\ 768\}$ – elles ont leur propre table de Huffman (voir tableau 2.4), avec, également, la possibilité de bits supplémentaires. Les symboles sont codés sur cinq bits : la distance 40 est codée `01010|0111`.

Deux options sont alors possibles : une statique et une dynamique. La version statique définit *au préalable* les tables de Huffman, précédemment décrites (voir tableaux 2.3 et 2.4), au sein du codeur et du décodeur.

Dans la version dynamique, la table est construite par le codeur qui la transmet au décodeur. L'intérêt de la version dynamique est d'améliorer le taux de compression du

5. Un caractère = un octet.

Symboles	Bits	Longueurs	Symboles	Bits	Longueurs
257	0	3	271	2	27–30
258	0	4	272	2	31–34
259	0	5	273	3	35–42
260	0	6	274	3	43–50
261	0	7	275	3	51–58
262	0	8	276	3	59–66
263	0	9	277	4	67–82
264	0	10	278	4	83–98
265	1	11, 12	279	4	99–114
266	1	13, 14	280	4	115–130
267	1	15, 16	281	5	131–162
268	1	17, 18	282	5	163–194
269	2	19–22	283	5	195–226
270	2	23–26	284	5	227–257
			285	0	258

Tableau 2.2. *Les longueurs sont codées à l'aide d'un alphabet de symboles. Certains symboles représentent plusieurs longueurs. Aussi, pour distinguer les longueurs associées à un même symbole, un ou plusieurs bits sont adjoints au symbole*

Symboles	Taille en bits	Codes
0–143	8	`00110000` $\cdots$ `10111111`
144–255	9	`110010000` $\cdots$ `111111111`
256–279	7	`0000000` $\cdots$ `0010111`
280–287	8	`11000000` $\cdots$ `11000111`

Tableau 2.3. *Table de Huffman utilisée pour le codage des symboles*

Symboles	Bits	Distances	Symboles	Bits	Distances
0	0	1	15	6	193–256
1	0	2	16	7	257–384
2	0	3	17	7	385–512
3	0	4	18	8	513–768
4	1	5, 6	19	8	769–1024
5	1	7, 8	20	9	1025–1536
6	2	9–12	21	9	1537–2048
7	2	13–16	22	10	2049–3072
8	3	17–24	23	10	3073–4096
9	3	25–32	24	11	4097–6144
10	4	33–42	25	11	6145–8192
11	4	49–64	26	12	8193–12288
12	5	65–96	27	12	12289–16384
13	5	97–128	28	13	16385–24576
14	6	129–192	29	13	24577–32768

Tableau 2.4. *Codage des distances*

bloc en s'approchant de l'entropie par une étude statistique : fréquences d'apparition des lettres, des longueurs et des distances au sein du bloc.

Comme une étude statistique peut générer plusieurs arbres équivalents suivant l'ordre dans lequel les *minima* égaux sont traités, la technique de déflation opère une normalisation des arbres suivant les règles suivantes :

1) les codes apparaissent dans l'ordre croissant des longueurs ;

2) les codes de longueurs binaires identiques sont rangés dans l'ordre lexicographique.

EXEMPLE 2.1.– *Prenons l'alphabet $\{A, B, C, D\}$ dont l'arbre est montré en figure 2.8 et dont la table générée est donnée par le tableau 2.5 [DEU 96]. De cet alphabet et des longueurs de chaque code, on déduit que le symbole B est rangé en premier, puis vient le symbole A, suivi des symboles C et D. La table résultante est donnée par le tableau 2.6.*

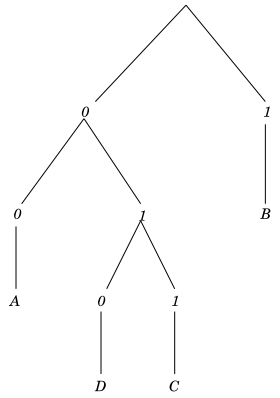


Figure 2.8. *Arbre de Huffman généré pour l'alphabet $\{A, B, C, D\}$*

Symbole	A	B	C	D
Code	00	1	011	010

Tableau 2.5. *Table générée à partir de l'arbre décrit en figure 2.5*

Symbole	A	B	C	D
Code	10	0	110	111

Tableau 2.6. *Normalisation de la table*

Cette normalisation permet au décodeur de reconstruire l'arbre de Huffman similaire à celui généré par le codeur en ne connaissant que l'alphabet initial et la longueur de chaque code.

Pour l'exemple précédent, le codeur envoie la séquence de longueurs $\{2, 1, 3, 3\}$ qui est mise en correspondance avec l'alphabet $\{A, B, C, D\}$ par le décodeur [DEU 96] :

1) il y a exactement un et un seul symbole de longueur 1 ; B est donc codé `0` ;

2) il y a également un seul code de longueur 2 ; donc, A est codé `10` ;

3) il y a deux symboles de longueur 3 ; l'ordre lexicographique étant respecté, C est codé `110` et D `111`.

Pour augmenter le taux de compression, les séquences de longueurs de la table de Huffman des longueurs et des lettres et de la table de Huffman des distances sont à leur tour codées suivant l'algorithme de Huffman.

L'alphabet des séquences des longueurs est le suivant :

1) les codes 0 à 15 sont utilisés pour les longueurs de 0 à 15 où 0 indique que le code n'est pas utilisé ;

2) le code 16 indique que la longueur précédente est copiée entre trois et six fois ; les 2 bits qui suivent informent sur le nombre exact de copies ;

3) le code 17 indique une répétition de la longueur 0 entre trois et dix fois ; les 3 bits qui suivent informent sur le nombre exact de copies ;

4) le code 18 indique une répétition de la longueur 0 entre onze et 138 fois ; les 7 bits qui suivent informent sur le nombre exact de copies.

2.2.4. *Synthèse du format PNG*

Ce format tend à remplacer le format GIF pour les images fixes de dimensions conséquentes, et dont le nombre de bits par pixel est au moins égal à huit. Toutefois, il ne permet pas le codage d'images animées contrairement au format GIF; il semblerait que sa version animée, MNG, ne soit guère utilisée.

Son principal avantage est d'utiliser les outils de compression habituellement embarqués avec les systèmes d'exploitation, à savoir les techniques de déflation/inflation.

2.3. LossLess JPEG

Le codeur JPEG (voir chapitre 3) propose plusieurs modes dont un sans perte : *LossLess* JPEG. L'algorithme du mode sans perte est assez simple. Il est résumé en figure 2.9. Le processus traite indépendamment chaque composante de l'image.

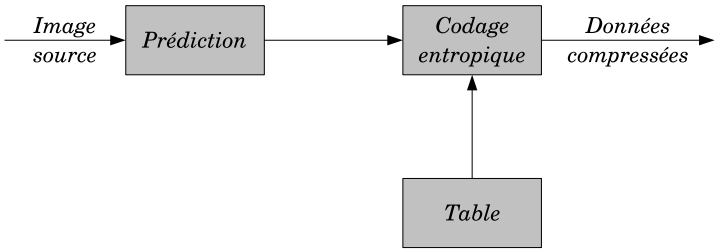


Figure 2.9. *Schéma algorithmique du mode LossLess du format jpeg*

Pour chaque composante, le traitement est séquentiel. Chaque échantillon p est prédit. La différence avec sa prédiction $\widehat{\tilde{p}}$ est codée : $\epsilon = p - \widehat{\tilde{p}}$. Le codage est entropique de type Huffman ou arithmétique. Dans le cas d'un codage arithmétique, la version QM est utilisée (voir annexe B volume 1).

La prédiction est effectuée sur un voisinage direct illustré en figure 2.10. Pour le pixel courant p à coder, ses voisins sont ceux déjà codés. Ainsi, les valeurs utilisées pour la prédiction sont les valeurs reconstruites $\hat{a}$, $\hat{b}$ et $\hat{c}$. De cette manière, le codeur et le décodeur ont une prédiction similaire.

$\hat{c}$	$\hat{b}$
$\hat{a}$	p

Figure 2.10. *Le voisinage de prédiction du pixel p*

Il existe huit choix de prédiction décrit dans le tableau 2.7. Le choix 0 signifie qu'aucune prédiction n'est faite. Il est utilisé par le mode hiérarchique du format JPEG qui sera vu dans le prochain chapitre. Les choix 1, 2 et 3 sont monodimensionnels, tandis que les choix 4, 5, 6 et 7 sont bidimensionnels.

Choix	Prédiction
0	aucune prédiction
1	$\widehat{\tilde{p}} = \hat{a}$
2	$\widehat{\tilde{p}} = \hat{b}$
3	$\widehat{\tilde{p}} = \hat{c}$
4	$\widehat{\tilde{p}} = \hat{a} + \hat{b} - \hat{c}$
5	$\widehat{\tilde{p}} = \hat{a} + (\hat{b} - \hat{c})/2$
6	$\widehat{\tilde{p}} = \hat{b} + (\hat{a} - \hat{c})/2$
7	$\widehat{\tilde{p}} = (\hat{a} + \hat{b})/2$

Tableau 2.7. *Les prédictions possibles*

Pour la première ligne de la composante, le choix 1 est obligatoire. Pour le tout premier échantillon de cette première ligne, la prédiction est fixée à 2^P, où P est une constante enregistrée dans l'entête du flux compressé.

Pour les lignes suivantes, la prédiction initialement choisie[6] est utilisée, sauf pour le premier échantillon de chaque ligne qui utilise obligatoirement le choix 2.

Le codage de Huffman travaille avec des valeurs sur 16 bits.

Le codeur arithmétique binaire QM, utilise les statistiques du voisinage 2D des échantillons.

Le tableau 2.8 donne les valeurs de *contexte*, κ, indexées par les valeurs des différences $D_b = p - b$ et $D_a = p - a$. La valeur de *décision* est toujours nulle : $\delta = 0$.

	0	+S	-S	+L	-L
0	0	4	8	12	16
+S	20	24	28	32	36
-S	40	44	48	52	56
+L	60	64	68	72	76
-L	80	84	88	92	96

– le symbole 0 indique une différence nulle ;
– le symbole +S indique une différence égale à +1 ou +2 ;
– le symbole -S indique une différence égale à -1 ou -2 ;
– le symbole +L indique une différence $\geqslant +3$;
– le symbole -L indique une différence $\leqslant -3$.

Tableau 2.8. *Les 25 contextes dépendent des différences $D_b = p - b$ et $D_a = p - a$. La première ligne et la première colonne indiquent les valeurs prises par les différences D_b et D_a respectivement*

2.3.1. *Synthèse du format LossLess JPEG*

Il faut tout d'abord noter que les autres modes de codage sont avec pertes et utilisent la DCT. Ceci est le principal attrait de ce format. Mais, puisque le mode sans perte ne l'utilise pas, il se trouve être un mode à part au sein du standard.

Le dernier chapitre de cette partie montre que ce n'est pas forcément le cas pour d'autres standards qui présentent une forte homogénéité algorithmique quel que soit le mode utilisé.

6. Egalement indiquée dans l'en-tête du flux compressé.

Synthétiquement, le mode *LossLess* JPEG conduit, principalement, à coder l'erreur de prédiction – qu'elle soit 1D ou 2D – à l'aide du codeur arithmétique QM ou du codeur de Huffman.

Du fait de la forte corrélation des échantillons, la distribution de l'erreur de prédiction est centrée, à forte décroissance. Elle permet de compresser sensiblement la source, sans engendrer de pertes. Cependant, les taux de compression généralement atteint ne sont pas très élevés.

Une solution, pour augmenter ces taux de compression, est de prendre en compte un voisinage plus large : si un pixel est situé au sein d'une région homogène, un codeur par plage sera alors optimal. Dans les autres cas, le voisinage permettra de définir un *contexte* de codification. L'algorithme de codage pourra alors opérer une adaptation de ses paramètes en fonction du contexte. C'est ce que propose de faire le standard JPEG-LS maintenant présenté.

2.4. JPEG-LS

Son algorithme, intitulé LOCO-I, pour *LOw COmplexity LOssleess COmpression for Images*, est dû à Marcelo J. Weinberger, Gadiel Seroussi et Guillermo Sapiro [WEI 96, WEI 99]. Il permet la compression des images à *niveaux de gris* et *couleurs*. Dans ce dernier cas, les composantes sont traitées indépendamment les unes des autres.

Cette section ne présente que la version niveaux de gris; pour les images couleurs on invite le lecteur intéressé à consulter l'article de M. J. Weinberger *et al.* [WEI 99].

La lecture de l'image est suppposée séquentielle, ligne par ligne, comme pour le fac-similé. Ainsi, les échantillons (pixels de l'image à coder) ne sont pas définis en 2D mais comme un signal 1D (le temps est imposé par l'ordre de lecture des échantillons).

Le principe repose sur les trois points suivants :

– une *étape de prédiction* de l'échantillon x_t délivre la valeur $\hat{x}_t$ estimée à partir d'un ensemble de n valeurs reconstruites[7], $\{\tilde{x}_{t-n}, \cdots, \tilde{x}_{t-1}\}$, de pixels déjà codés ;

– une *étape de modélisation* du *contexte* de x_t construit un indice contextuel à l'aide de l'ensemble des valeurs reconstruites ;

– une étape de calcul de l'*erreur de prédiction* $\epsilon_t = x_t - \hat{x}_t$ dont la spécificité est la prise en compte du *contexte* de x_t.

7. Comme souvent (voir chapitre 3 volume 1), la prédiction se fait à partir des reconstructions des valeurs précédemment codées. Le schéma de reconstruction correspond à celui suivi par le décodeur. Ceci afin de limiter les erreurs de prédiction.

L'originalité de l'approche repose sur l'utilisation conjointe d'une *prédiction* et d'un *contexte* pour estimer l'erreur de prédiction. Cette combinaison a pour objectif d'augmenter le taux de compression sans qu'il y ait pertes. Comme le disent les auteurs, il s'agit d'« enjoying the best of both worlds ». Le schéma général de l'algorithme est donné en figure 2.11.

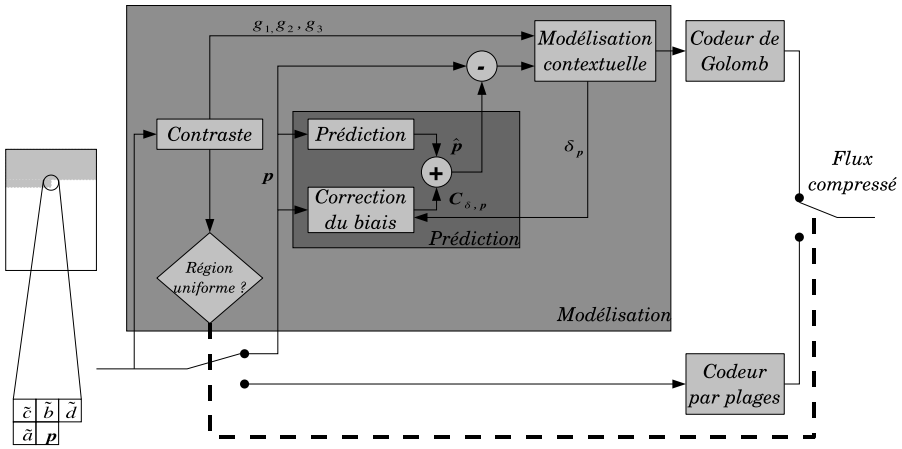


Figure 2.11. *Schéma général du codeur JPEG-LS. Pour alléger la description, l'étape de reconstruction, à l'aide du flux compressé, des échantillons $\tilde{a}, \tilde{b}, \tilde{c}$ et $\tilde{d}$, n'est pas indiquée*

Une analyse de l'uniformité du voisinage autour de la valeur x est utilisée pour savoir s'il faut effectuer un codage en *mode par plage* ou un codage *en mode régulier*. Si le voisinage est similaire à la valeur x, le codeur par plage est tout à fait adapté. Sinon, la forme de ce voisinage est prise en compte pour la prédiction et la modélisation du contexte. Ensuite, le codeur de Golomb (décrit das le chapitre 3 du volume 1) est utilisé.

Le paragraphe 2.4.1, présente :

– le module d'estimation du contraste dans le voisinage de la valeur x ;

– le test de planarité de ce voisinage ;

– le choix du mode qui en découle.

Les quatre paragraphes suivant décrivent le mode régulier. Le paragraphe 2.4.2 décrit la prédiction. Ensuite, le paragraphe 2.4.3 explique la modélisation du contexte. La correction du biais est décrite au paragraphe 2.4.4. Enfin, le paragraphe 2.4.5 présente l'adaptation contextuelle de l'algorithme de Golomb.

Le mode de codage par plages est décrit au paragraphe 2.4.6.

2.4.1. *Le contraste et les modes*

Le contraste est grossièrement estimé sous la forme de trois valeurs :

$$\begin{cases} g_1 &= \tilde{d} - \tilde{b} \\ g_2 &= \tilde{b} - \tilde{c} \\ g_3 &= \tilde{c} - \tilde{a} \end{cases}$$

où $\tilde{x}$ est la valeur reconstruite de x.

Ces trois valeurs informent sur le voisinage de la valeur du pixel p à coder. Chaque g_i indique s'il y a eu un changement dans les valeurs du voisinage. Ainsi, si $g_i = 0 \forall i$, alors la région est supposée uniforme et le mode par plage est déclenché. Sinon le mode est régulier.

2.4.2. *La prédiction*

La prédiction est estimée par détection des frontières, appellées *contours*, entre régions homogènes en intensité. Les trois modes possibles sont fondés sur les valeurs reconstruites des pixels $\tilde{a}$, $\tilde{b}$ et $\tilde{c}$ (voir figure 2.11)[8] :

1) si l'intensité du pixel $\tilde{c}$ est supérieure aux deux autres, le pixel p est alors supposé appartenir à un contour sombre horizontal ou vertical. Sa prédiction $\hat{p}$, est la valeur minimale entre les pixels $\tilde{a}$ et $\tilde{b}$. Ces cas sont illustrés en figures 2.12a et 2.12b :

si $\tilde{c} \geqslant \max(\tilde{a}, \tilde{b})$ **alors**

$$\hat{p} = \min(\tilde{a}, \tilde{b}) = \begin{cases} \tilde{b} & \text{(voir figure 2.12a)} \\ \tilde{a} & \text{(voir figure 2.12b)} \end{cases}$$

2) inversement, si la valeur du pixel $\tilde{c}$ est inférieure aux deux autres, le pixel p est supposé appartenir à un contour clair horizontal ou vertical. Sa prédiction $\hat{p}$, est la valeur maximale entre les pixels $\tilde{a}$ et $\tilde{b}$. Ces cas sont illustrés en figures 2.12c et 2.12d ;

si $\tilde{c} \leqslant \min(\tilde{a}, \tilde{b})$ **alors**

$$\hat{p} = \max(\tilde{a}, \tilde{b}) = \begin{cases} \tilde{b} & \text{(voir figure 2.12c)} \\ \tilde{a} & \text{(voir figure 2.12d)} \end{cases}$$

3) Dans tous les autres cas, le pixel $\tilde{c}$ a une valeur comprise entre celles de $\tilde{a}$ et de $\tilde{b}$. Ces trois pixels et leurs valeurs d'intensité forment un plan passant par les coordonnées $(x_a, y_a, \tilde{a})$, $(x_b, y_b, \tilde{b})$ et $(x_c, y_c, \tilde{c})$. La prédiction $\hat{p}$, est alors la valeur

8. Contrairement à l'estimation du contraste (et plus tard à la modélisation du contexte), le pixel $\tilde{d}$ n'est pas pris en compte.

de coordonnées (x_p, y_p, λ_p) qui est le symétrique de $\tilde{c}$, dans le plan, par rapport à la droite passant par les valeurs $\tilde{a}$ et $\tilde{b}$:

$$\lambda_p = \frac{1}{2}(\tilde{a} + \tilde{b}) + \left(\frac{1}{2}(\tilde{a} + \tilde{b}) - \tilde{c}\right)$$

La figure 2.12c illustre cette situation.

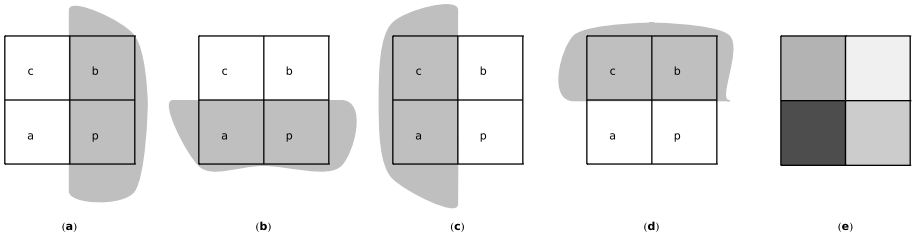


Figure 2.12. *Les trois modes de prédiction (d'après [TAU 02])*

En résumé, la prédiction correspond à l'équation suivante :

$$\hat{p} = \begin{cases} \min(\tilde{a}, \tilde{b}) & \text{si } \tilde{c} \geqslant \max(\tilde{a}, \tilde{b}) \\ \max(\tilde{a}, \tilde{b}) & \text{si } \tilde{c} \leqslant \min(\tilde{a}, \tilde{b}) \\ \tilde{a} + \tilde{b} - \tilde{c} & \text{sinon} \end{cases}$$

2.4.3. *Le contexte*

Le contexte est fonction des vecteurs de contraste (g_1, g_2, g_3). Ces vecteurs sont quantifiés en régions uniformes indicées entre $-T$ et T. Cette quantification partitionne l'espace des vecteurs en $(2T + 1)^3$ régions. En considérant la symétrie de la distribution de l'erreur de prédiction (voir figure 2.13), cet ensemble peut être réduit de moitié :

$$P\left(X_e = \Delta | (g_1, g_2, g_3)\right) = P\left(X_e = -\Delta | (-g_1, -g_2, -g_3)\right)$$

Ainsi, le nombre de contextes finalement retenu vaut :

$$4T^3 + 6T^2 + 3T + 1$$

Le standard JPEG-LS fixe T à quatre ce qui autorise 365 contextes. Les trois seuils de la quantification sont, par défaut, fixés à 3, 7 et 21. Mais, ils peuvent être définis par l'utilisateur et sont donc à transmettre dans le flux compressé final. L'ensemble des opérations de quantification est résumé dans l'algorithme qui suit où s_p informe sur le changement de signe.

```
MODÉLISATION DU CONTEXTE (g1, g2, g3)
 1  s_p ← 0
 2  pour j = 1, 2, 3
 3  faire si g_j = 0
 4          alors q_j ← 0
 5          sinonsi |g_j| < T1
 6                  alors q_j ← 1
 7                  sinonsi |g_j| < T2
 8                          alors q_j ← 2
 9                          sinonsi |g_j| < T3
10                                  alors q_j ← 3
11                                  sinon q_j ← 4
12        si s_p = 0
13          alors si g_j < 0
14                  alors s_p ← −1
15                  sinonsi g_j > 0
16                          alors s_p ← +1
17  si s_p = 0
18    alors s_p ← +1
19  Q_p ← 81q1 + 9q2 + 3q3  // contexte du pixel p
```

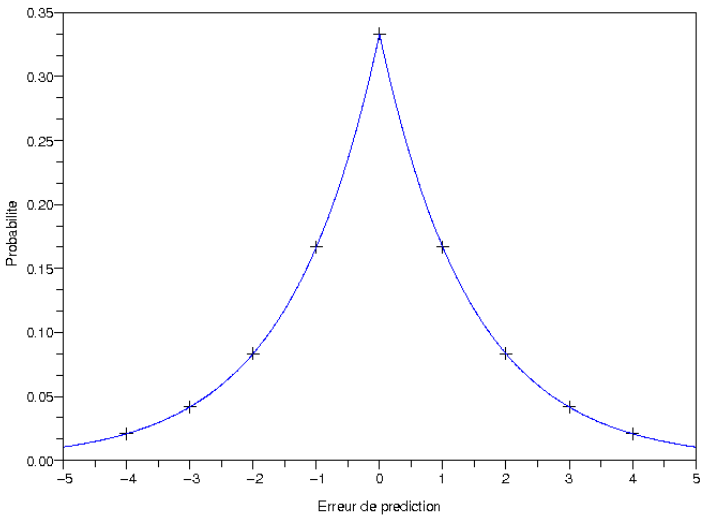


Figure 2.13. *La distribution X_e, de l'erreur de prédiction e_p offre une symétrie centrée à l'origine. Les abscisses sont les erreurs de prédiction e_p, et les ordonnées les probabilités*

2.4.4. *La correction du biais*

Si l'alphabet de l'image (c'est-à-dire les valeurs d'intensité autorisées) est de taille α, l'erreur à coder, $e_p = p - \hat{p}$, entre la prédiction et la valeur exacte est comprise entre $-\hat{p}$ et $(\alpha - \hat{p})$. Puisque le décodeur connaît également la prédiction $\hat{p}$, cet intervalle peut être ramené à :

$$[-\lfloor \alpha/2 \rfloor, \lceil \alpha/2 \rceil - 1]$$

Comme l'illustre la figure 2.13, e_p a une distribution symétrique à forte décroissance et centrée en l'origine. Cependant, cette distribution n'est exacte que pour des sources sans mémoire[9].

Dans le cas des images, le contexte est important; les pixels ne sont pas indépendants. La distribution reste laplacienne, mais, son centre subit un décalage. Ce phénomène est accentué par l'utilisation de valeurs *entières*.

Le centre de la distribution est donc translaté, d'une valeur réelle μ, qu'il faut corriger. Pour ce faire, μ est d'abord réécrit en une partie entière, appelée le *biais* R, et une partie réelle de *décalage* d, comprise entre -1 et 1.

$$\mu = R - d$$

Puisque le biais R, est une valeur entière, il peut être ajouté à $\hat{p}$ afin d'en éliminer les effets. Pour estimer ce biais, l'idéal serait de connaître la valeur médiane de la distribution. Mais, pour des raisons d'efficacité, la valeur moyenne est préférée :

$$C[Q_p] = \lceil \frac{D[Q_p]}{N[Q_p]} \rceil$$

où $D[Q_p]$ est la variable de cumul des erreurs de prédiction calculées jusque-là pour le contexte Q_p et $N[Q_p]$ est le compteur de ces erreurs de prédiction.

REMARQUE 2.2.– *Cette estimation de la moyenne est effectuée indépendamment pour chaque contexte Q_p.*

Ainsi, $C[Q_p]$ reflète l'estimation de la moyenne pour le contexte Q_p au moment de coder l'erreur e_p. Cette moyenne représente le biais dû à la prédiction :

$$\hat{p} = \hat{p} + s_p C[Q_p]$$

où s_p est la valeur du signe donné par l'analyse du contexte (voir l'algorithme du paragraphe 2.4.3).

9. C'est-à-dire sans relation entre les échantillons.

Toutefois, cette moyenne implique une division qui n'est pas souhaitable dans un (dé)codeur efficace. Elle est donc réécrite sous la forme suivante :

$$D[Q_p] = C[Q_p]N[Q_p] + B[Q_p]$$

avec $-N[Q_p] < B[Q_p] \leqslant 0$.

L'algorithme qui en découle ajoute l'erreur courante e_p au reste $B[Q_p]$, incrémente le compteur $N[Q_p]$, puis, effectue les corrections nécessaires :

```
CORRECTION_DU_BIAIS (e_p, Q_p)
 1  B[Q_p] ← B[Q_p] + e_p
 2  N[Q_p] ← N[Q_p] + 1
 3  si B[Q_p] ⩽ −N[Q_p]
 4     alors C[Q_p] ← C[Q_p] − 1
 5           B[Q_p] ← B[Q_p] + N[Q_p]
 6           si B[Q_p] ⩽ −N[Q_p]
 7              alors B[Q_p] ← −N[Q_p] + 1
 8     sinonsi B[Q_p] > 0
 9              alors C[Q_p] ← C[Q_p] + 1
10                    B[Q_p] ← B[Q_p] − N[Q_p]
11                    si B[Q_p] > 0
12                       alors B[Q_p] ← 0
```

Initialement, les variables $B[Q_p]$ et $C[Q_p]$ valent 0 tandis que $C[Q_p]$ vaut 1. Lorsque le reste $B[Q_p]$, sort de l'intervalle ($B[Q_p] \leqslant -N[Q_p]$ ou > 0), il est corrigé. En même temps le quotient $C[Q_p]$ est ajusté.

De plus, pour éliminer les effets des valeurs aberrantes[10], le reste $B[Q_p]$, est fixé à la borne la plus proche de son intervalle ($-N[Q_p]$ ou 0). Il est à noter qu'alors $\frac{B[Q_p]}{N[Q_p]}$ est l'estimation de la moyenne du décalage d.

2.4.5. *Algorithme de Golomb*

La symétrie autorise le regroupement des valeurs positives et négatives afin de diminuer de moitié le domaine de définition de la distribution. Aussi, le signe est pris

10. C'est-à-dire celles dont le reste ne peut être corrigé comme indiqué précédemment. Elles sont souvent dues à de mauvaises conditions de prédiction.

en compte séparément et le codage est alors de la forme :

$$e_p = s_p.(p - \hat{p})$$

avec $s_p = -1$ ou $+1$.

Ceci revient à *plier* la courbe de la distribution au niveau de sa valeur centrale (0) pour ne fournir que des valeurs positives d'erreur. Sa forme devient alors géométrique avec une unique pente décroissante. Ainsi, l'algorithme de Golomb peut être utilisé pour un codage optimal.

Mais, bien que le biais R, ait été éliminé, la distribution est toujours sujette au décalage d compris entre -1 et 0. La figure 2.14 illustre cette situation pour $d = -0,7$ et $d = -0,3$.

Dans ce cas, prendre directement les valeurs absolues des erreurs de prédiction ne décrit plus une distribution monotone décroissante. JPEG-LS opère donc une transformation des erreurs afin de conserver la décroissance monotone de la distribution.

Deux situations se présentent : soit la moyenne est inférieure à $-\frac{1}{2}$; soit elle est supérieure à cette valeur. Le cas où $d \leqslant -\frac{1}{2}$ revient à écrire $e_p \leftarrow -(e_p + 1)$. La transformation ne concerne donc que le cas où $d > -\frac{1}{2}$:

$$\bar{e}_p = 2|e_p| - \mathrm{neg}(e_p)$$

avec :

$$\mathrm{neg}(y) = \begin{cases} 0 & \text{si } y \geqslant 0 \\ 1 & \text{sinon} \end{cases}$$

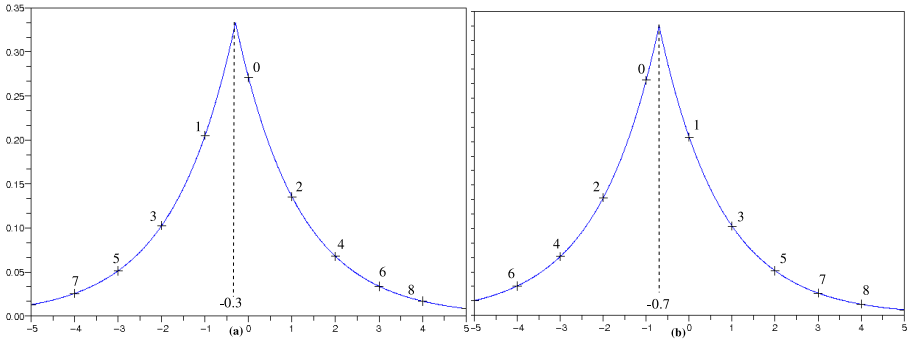


Figure 2.14. *Introduction du décalage d dans la distribution de l'erreur de prédiction : (a) $d = -0,3$; (b) $d = -0,7$. Les valeurs des transformées $\bar{e}_p$ sont inscrites directement sur la courbe de la distribution*

La figure 2.14 montre l'entrelacement des valeurs e_p quand les valeurs $\bar{e}_p$ vont en décroissant.

Pour coder l'erreur de prédiction $\bar{e}_p$, l'algorithme de Golomb, décrit dans le chapitre 3 du volume 1, a besoin d'estimer le paramètre k :

$$k = \max\left(0, \left\lceil \log_2\left(\frac{E(X_{\bar{e}})}{2}\right)\right\rceil\right)$$

où $X_{\bar{e}}$ est la VA de l'erreur de prédiction.

L'espérance $E(X_{\bar{e}})$ n'est évidemment pas connue et doit donc être ajustée à chaque nouveau pixel p codé. L'algorithme de Golomb (voir chapitre 3 volume 1) opère cet ajustement en utilisant un compteur N et une variable de cumul A. Ce principe est modifié pour prendre en compte le contexte δ_p et les valeurs transformées des erreurs $\bar{e}_p$. Quatre variables sont alors utilisées : $A[Q_p]$, $N[Q_p]$, $B[Q_p]$ et $C[Q_p]$. Leurs indices expriment la dépendance contextuelle.

Ces variables servent au calcul :

– du rapport $\frac{B[Q_p]}{N[Q_p]}$ qui est l'estimation courante $E(X_e|Q_p)$ de l'espérance des erreurs de prédiction e_p connaissant le contexte $Q_p \Longrightarrow$ il est utilisé pour le calcul de la transformée $e_p \mapsto \bar{e}_p$;

– du rapport $\frac{A[Q_p]}{N[Q_p]}$ qui est l'estimation $E(X_{\bar{e}}|Q_p)$ des valeurs transformées des erreurs $\bar{e}_p$ connaissant le contexte $Q_p \Longrightarrow$ il est utilisé pour estimer la valeur k de l'algorithme de Golomb (voir chapitre 3 volume 1).

La variable $C[Q_p]$ enregistre la valeur du biais introduit dans les erreurs de prédiction $e_p \Longrightarrow$ elle est utilisée pour la correction du biais. Initialement, $A[Q_p] = \max(2, \lfloor(\alpha + 32)/64\rfloor)$ [WEI 99].

Toujours dans un soucis d'éviter l'opération de division, l'estimation du paramètre k peut également être réécrite :

$$k = \max\left(0, \lceil \log_2 \frac{A[Q_p]}{N[Q_p]} \rceil\right) = \underset{k\in\mathbb{N}^+}{\operatorname{argmin}} \left(2^k N[Q_p] > A[Q_p]\right)$$

2.4.6. *Mode par plage*

Le codage de Golomb souffre du même inconvénient que l'algorithme de Huffman. Lorsque l'entropie est faible, l'algorithme ne pouvant descendre en deçà d'un bit par symbole, le codage n'est pas optimal.

Typiquement, les régions uniformes en intensité de l'image à coder ont une faible entropie. Aussi, une extension de l'alphabet de la source est envisagée pour ses zones

particulières. Ce procédé est similaire à celui utilisé par le codeur de Huffman : il s'agit de construire des *hypersymboles* regroupant plusieurs échantillons. Le passage du mode normal ou mode par plage est effectué grâce au test d'uniformité du contexte : $[g_1, g_2, g_3] = [0, 0, 0]$. Autrement dit, $\tilde{a} = \tilde{b} = \tilde{c} = \tilde{d}$.

Une fois en mode par plage, le processus suppose qu'il existe une séquence de valeurs identiques à $\tilde{a}$. Cette séquence s'arrête quand la fin de la ligne est atteinte ou lorsqu'une valeur diffère de $\tilde{a}$. La longueur de la séquence est alors codée. Le décodeur n'a pas besoin de connaître la valeur $\tilde{a}$ puisqu'il se trouvera, au même instant, dans le même contexte et effectuera, donc, le même changement de mode. Lorsque la séquence s'arrête avec une fin de ligne, le codeur revient directement en mode régulier. En revanche, quand une valeur $x \neq \tilde{a}$ vient arrêter la séquence, le codeur entre en état *end-of-run*.

La longueur r, de la séquence est codée à l'aide du paramètre m – détaillé ci-après – et d'un codage unaire inversé : le bit `1` indique le codage d'une portion – de taille $\leqslant m$ – d'une séquence tandis que le bit `0` tient lieu de séparateur entre les séquences. L'alphabet utilisé par ce codage est fini : $\{0, 10, 110, \cdots, 1^{m-1}0, 1^m\}$. Mais les longueurs r, des séquences ne sont pas forcément des multiples de m. Aussi, un bit `0` est suivi par le code binaire de la longueur de la fin de la séquence. Ainsi, l'algorithme de Golomb d'ordre $m = 2^k$ est optimal pour ce codage. Toutefois, lors du codage d'une séquence, l'ordre k est ajusté grâce à une table T et un indice I_{run}. L'indice référence l'entrée dans la table qui permet de mettre à jour la valeur de k. Initialement, l'indice $I_{\text{run}} = 0$. Il est incrémenté (resp. décrémenté) à chaque émisssion d'un bit `1` (resp. d'un bit `0`). Le procédé est décrit par l'algorithme qui suit :

```
CODEUR PAR PLAGE()
 1  k ← T[I_run]  // retrouver la valeur de k dans la table T
 2  m ← 2^k tant que r ⩾ m
 3  faire Emettre 1 // la sous-séquence de taille m est codée
 4        r ← r − m  // mise à jour de la longueur à coder
 5        I_run ← min(I_max, I_run + 1)  // l'indice est incrémenté
 6        k ← T[I_run]
 7        m ← 2^k  // mise à jour en cours de codage de la séquence
 8  si END_OF_RUN
 9    alors  // la séquence s'arrête avant la fin de ligne
10          Emettre 0 // fin de la séquence
11          Emettre k bits // pour coder la fin de la séquence
12          I_run ← max(0, I_run − 1)
13    sinon  // fin de ligne rencontré
14          si r > 0
15            alors Emettre 1 // la longueur à coder est ⩽ m
```

La table T vaut :

0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3

4	4	5	5	6	6	7	7	8	9	10	11	12	13	14	15

Lorsque la séquence se termine par une valeur x différente de la valeur de référence $\tilde{a}$, la différence de cet échantillon avec son voisin direct, $\tilde{b}_x$, est codée. L'algorithme de Golomb est utilisé comme en mode régulier [PNG 03].

2.4.7. *Quasi sans perte*

Le standard JPEG-LS autorise un codage quasi sans perte grâce à un paramètre de contrôle NEAR que l'utilisateur peut ajuster. Si $\mathrm{NEAR} = 0$, le codage est sans perte. Sinon $(\mathrm{NEAR} > 0)$, le calcul de l'erreur e_p et la reconstruction $\tilde{x}$ sont à adapter :

$$\begin{aligned} e_p &\leftarrow \mathrm{sign}(e_p)\frac{\mathrm{abs}(e_p) + \mathrm{NEAR}}{2\,\mathrm{NEAR}+1} \\ \tilde{x} &\leftarrow x + s_p e_p(2\,\mathrm{NEAR}+1) \end{aligned}$$

La substitution ne change pas le comportement de l'algorithme puisque le codage sans perte correspond au cas où $\mathrm{NEAR} = 0$. Certains autres changements mineurs sont également nécessaires. Par exemple, le choix du mode par plage se fera quand le test suivant est vérifié :

$$\mathrm{abs}(g_1) < \mathrm{NEAR} \text{ et } \mathrm{abs}(g_2) < \mathrm{NEAR} \text{ et } \mathrm{abs}(g_3) < \mathrm{NEAR}$$

2.4.8. *Synthèse du format JPEG-LS*

Ce standard propose une adaptation contextuelle forte pour obtenir des taux de compression élevés. Le contexte est défini par le voisinage du pixel observé. Il intervient dans le choix du mode de compression : régulier ou par plages.

Si le voisinage est contrasté, le codeur entre en mode régulier. Ce voisinage est utilisé à la fois pour estimer la prédiction et pour modéliser le contexte associé au pixel à coder.

La prise en compte du voisinage par la prédiction permet de diminuer l'entropie de la source à coder. La distribution de la prédiction est alors de type laplacienne, c'est-à-dire symétrique et centrée en 0. En repliant cette courbe en son centre, cette distribution devient géométrique. Pour effectuer ce repliement, JPEG-LS s'assure que

le centre est bien en 0 par une étape de traitement de la source appelée *correction du biais*.

L'algorithme de Golomb étant optimal pour des distributions géométriques, il est utilisé avec, en complément, une modélisation contextuelle. Celle-ci opère une quantification des données qui permet de réduire l'alphabet à coder à 365 entrées.

Lorsque le voisinage est uniforme (ou presque en mode quasi sans perte) un codage par plages est effectué afin de regrouper toutes les valeurs identiques en un unique symbole (la longueur de la plage). A nouveau, l'algorithme de Golomb est utilisé.

Ainsi, il existe deux modes comportant des prétraitements différents mais utilisant le même codeur entropique en fin de chaîne.

2.5. Synthèse

Ce chapitre vient de présenter quelques-uns des formats de compression sans perte parmi les plus connus. En conclusion immédiate, il apparaît qu'aucun de ces formats n'est supérieur aux autres en toute situation. Quand le format GIF s'adapte bien aux imagettes de synthèse tels que les logos, le format PNG propose une compression plus performante pour les images de plus grandes tailles.

Toutefois, ces deux formats, GIF et PNG, sont limités en taux de compression dans le cas des images naturelles (photographies). Ici, les formats *LossLess* JPEG et JPEG-LS prennent l'avantage. Ils offrent des taux de compression similaires (voir figure 2.15), supérieurs aux précédents formats.

LossLess JPEG est un mode particulier du standard JPEG. Son algorithme repose sur la prédiction adaptative : huit choix de prédiction sont possibles. Il est à noter que le mode *LossLess* est très différent des autres modes du standard JPEG qui utilisent le découpage de l'image en blocs et la transformée en cosinus discrète.

En revanche, JPEG-LS n'est pas un mode du standard JPEG. C'est un outil indépendant qui cherche à augmenter les taux de compression en observant le voisinage des pixels à coder. Ce voisinage sert à différentes étapes de l'algorithme :

– il définit le choix du mode de codage, régulier ou par plages ;

– il intervient dans l'estimation de la prédiction ;

– il permet de modéliser le contexte du pixel en cours de codage.

De ce fait, la prédiction est comparable à celle du format *LossLess* JPEG. Mais, une modélisation contextuelle vient s'ajouter afin de quantifier les valeurs à coder. Cette modélisation participe, donc, à la diminution de l'entropie de la source à coder.

Dans le prochain chapitre, les autres modes du standard JPEG sont présentés en détail.

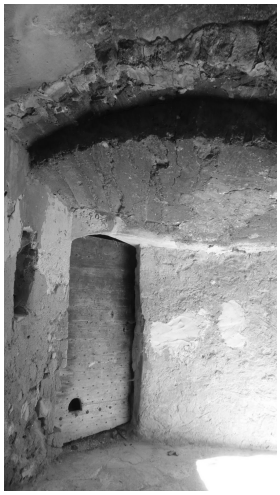


Figure 2.15. *Comparatif des format : l'image présentée occupe 1,4 Mo au format BMP (format bitmap non compressé de Windows) alors qu'elle n'occupe que 701,2Ko au format JPEG-LS. C'est une réduction de moitié de l'espace mémoire nécessaire sans aucune perte*

Chapitre 3

JPEG : *Joint Photographic Expert Group*

L'objectif du standard JPEG [PEN 93] est de transformer une image, pouvant contenir plusieurs composantes, en un flux binaire adapté à la diffusion sur Internet. Pour cela, le standard définit une compression qui engendre des pertes contrôlées. Celles-ci sont aussi peu perceptibles que possibles. Evidemment, plus le taux de compression est élevé, plus les pertes engendrées sont visibles.

L'image est structurée comme suit. Le nombre de colonnes x_i et le nombre de lignes y_i de la composante, indicée par i, sont codés :

$$x_i = \left\lceil X * \frac{H_i}{H_{\max}} \right\rceil \text{ et } y_i = \left\lceil Y * \frac{V_i}{V_{\max}} \right\rceil$$

où $H_{\max}$ et $V_{\max}$ sont les facteurs d'échelle maximaux pour l'ensemble des composantes et où X et Y sont les dimensions de l'image dans sa globalité : c'est-à-dire X est le maximum des x_i de toutes les composantes et Y le maximum des y_i.

EXEMPLE 3.1.– *Pour une image de 512 lignes de 512 pixels chacune, ayant trois composantes de facteur d'échelle respectif :*

composante 0	$H_0 = 4$	$V_0 = 1$
composante 1	$H_1 = 2$	$V_1 = 2$
composante 2	$H_2 = 1$	$V_2 = 1$

$X = 512$, $Y = 512$, $H_{\max} = 4$, $V_{\max} = 2$ *et les* x_i *et* y_i *des composantes valent :*

composante 0	$x_0 = 512$	$y_0 = 256$
composante 1	$x_1 = 256$	$y_1 = 512$
composante 2	$x_2 = 128$	$y_2 = 256$

Le standard JPEG procède en quatre grandes étapes. Premièrement, une mise en forme des pixels est faite. Ceux-ci sont regroupés en blocs disjoints de taille 8×8. Puis, ces blocs sont transformés pour une analyse fréquentielle. La troisième étape récupère les coefficients de cette analyse et les quantifie. Cette quantification engendre des pertes (voir chapitre 3 du volume 1). Enfin, la dernière étape effectue le codage entropique – sans perte – des valeurs quantifiées. La figure 3.1a montre le schéma global d'un codeur JPEG.

Le décodeur opère les mêmes étapes, mais, en sens inverse (voir figure 3.1b). Les données compressées sont tout d'abord décodées pour être ensuite déquantifiées. Les valeurs obtenues sont alors des approximations des coefficients de l'analyse fréquentielle du codeur. La transformation inverse de celle du codeur est appliquée pour retrouver les valeurs des pixels. Bien que la transformation inverse soit exacte (sans perte), les pixels reconstruits sont des approximations des pixels de la source.

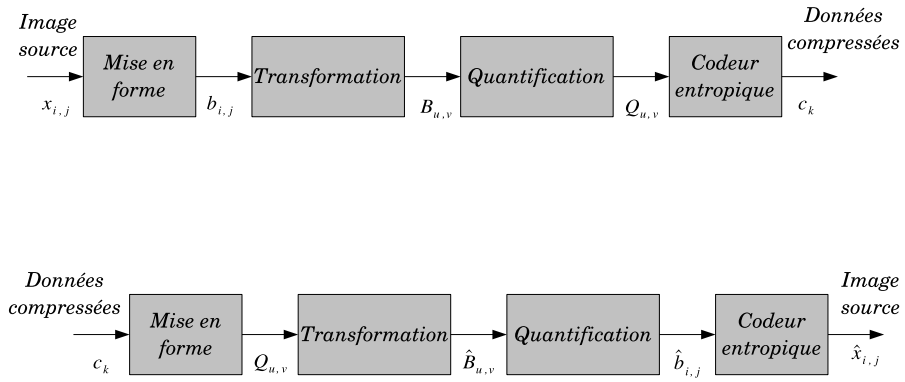


Figure 3.1. *Schémas généraux (a) du codeur et (b) du décodeur JPEG*

Pour éviter les pertes, il existe un mode particulier déjà vu au chapitre 2 : le mode *LossLess*-JPEG. Dans ce mode, aucune quantification n'est effectuée et donc aucune perte n'est engendrée. De plus, c'est le seul mode qui opère un codage différentiel (DPCM) au lieu de faire une analyse fréquentielle.

Les autres modes que ce chapitre va maintenant présenter sont au nombre de trois :

– le mode *séquentiel* est celui par défaut. Il est donc décrit en premier dans la section 3.1 ;

– le mode *progressif* est une modification du principe d'affichage des images pour en accélérer la visualisation lors de leurs décompressions. Mais, en dehors de cela, il reste identique au mode séquentiel. La section 3.2 présente la modification proposée ;

– le dernier mode est *hiérarchique*. Son objectif est également d'accélérer l'affichage des images à la décompression. Cependant, son approche modifie la structure de représentation de l'image. Le principe est de la famille des analyses multirésolutions. Du fait même, il repose sur n'importe lequel des trois modes précédents : sans-perte, séquentiel ou progressif. La section 3.3 détaille cette approche.

3.1. Le mode séquentiel

Le schéma général, de la figure 3.1a, décrit parfaitement ce mode. La source est mise en forme pour être transformée, puis, quantifiée, avant d'être codée.

3.1.1. *La mise en forme*

L'image à compresser peut être constituée d'une seule composante. Il s'agit alors d'une image à niveau de gris, allant du noir au blanc, dont les valeurs des pixels sont généralement comprises entre 0 et 255. Cette unique composante est alors découpée en blocs disjoints de taille 8×8. Ces blocs sont codés indépendamment les uns des autres. Leur lecture se fait de haut en bas et de droite à gauche comme le montre la figure 3.2.

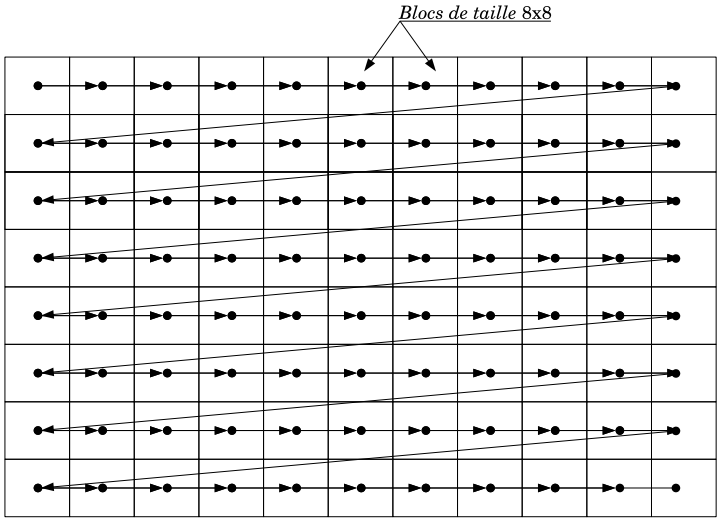


Figure 3.2. *Ordre de lecture des blocs d'une image monocomposante*

En revanche, lorsque l'image comporte plusieurs composantes, celles-ci peuvent être de tailles différentes. Le traitement peut se faire, soit indépendamment pour chacune des composantes, soit en mixant l'ordre d'acquisition des blocs des différentes composantes[1]. Dans tous les cas, les blocs sont toujours de taille 8×8.

Si chaque composante est traitée séparément des autres, l'acquisition des blocs à compresser se fait, composante après composante, de haut en bas et de droite à gauche, comme pour les images monocomposantes (voir figure 3.2).

Dans le cas de composantes mixées, un ordre doit être précisé. Celui-ci indique le nombre de blocs de la première composante à coder, suivi du nombre de blocs de la seconde composante et ainsi de suite jusqu'à la dernière composante. Cet ordre définit l'*unité de codage minimum*, d'abréviation MCU pour *Minimum Coded Unit*. Une unité regroupe donc l'ensemble des blocs à traiter lors de la même passe. M

Par exemple, prenons une image à trois composantes. La deuxième composante est sous-échantillonnée de 2:1 en colonnes par rapport à la première composante. La troisième composante est sous-échantillonnée de 2:1 en lignes. Le sous-échantillonnage de 2:1 signifie qu'un pixel sur deux est conservé. Aussi, la hauteur de la deuxième composante est la moitié de la hauteur de la première composante alors que la largeur de la troisième composante est la moitié de la largeur de la première composante. La MCU correspondante regroupe quatre blocs de la première composante, suivie de deux blocs de la deuxième et de deux blocs de la troisième. L'ordre de récupération de ces blocs est montré en figure 3.3.

En particulier, le cas des images couleurs est pris en compte. L'espace couleur proposé par le standard est l'espace YCrCb (voir chapitre 4 volume 1) avec un sous-échantillonnage de 2:1 en lignes et en colonnes des axes de chrominance Cr et Cb, par rapport à l'axe de luminosité Y. Il est possible d'ajuster le sous-échantillonnage des axes aussi bien en lignes qu'en colonnes. Cet ajustement peut varier entre un et quatre. Toutefois, cette option n'est guère acceptée par les codeurs et décodeurs développés par les industriels. La mise en forme permet de faire la conversion de l'espace RGB vers l'espace YCrCb. Le sous-échantillonnage de la chrominance par rapport à la luminosité est justifiée par l'importance de la luminosité dans le système visuel humain (voir chapitre 4 volume 1). Toutefois, l'espace couleurs n'étant pas imposé, le codeur peut choisir un autre espace que l'espace YCrCb. Il faut donc ajouter, dans l'en-tête du flux binaire, la conversion de l'espace choisi vers l'espace RGB pour que le décodeur puisse appliquer la transformée inverse.

1. En anglais les termes *no-interleaved components* et *interleaved component* sont utilisés et devraient être traduits par *composantes non entrelacées* et *composantes entrelacées*. Cependant, nous utiliserons les traductions *composantes indépendantes* et *composantes mixées* pour éviter toute ambiguïté avec la définition d'entrelacement utilisée en vidéo.

Une fois la conversion faite, les pixels sont regroupés en blocs (de taille 8×8) qui sont à leurs tours regroupés en MCU. Pour le mode couleur par défaut, la MCU regoupe quatre blocs *Y*, un *Cr* et un *Cb* comme le montre la figure 3.4 ; il y a identité entre MCU et bloc sur les composantes de chrominance.

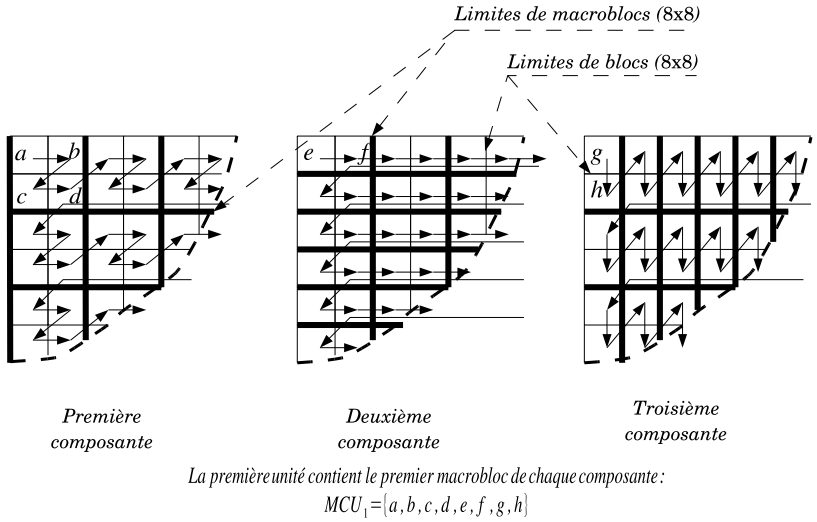


Figure 3.3. *Ordre de lecture d'une image à trois composantes telle que la deuxième soit un sous-échantillonnage de 2:1 en colonnes et la troisième un sous-échantillonnage de 2:1 en lignes. En trait pointillé gras, sont encadrés les blocs appartenant à une même MCU. Les blocs sont représentés par des traits fins continus.*

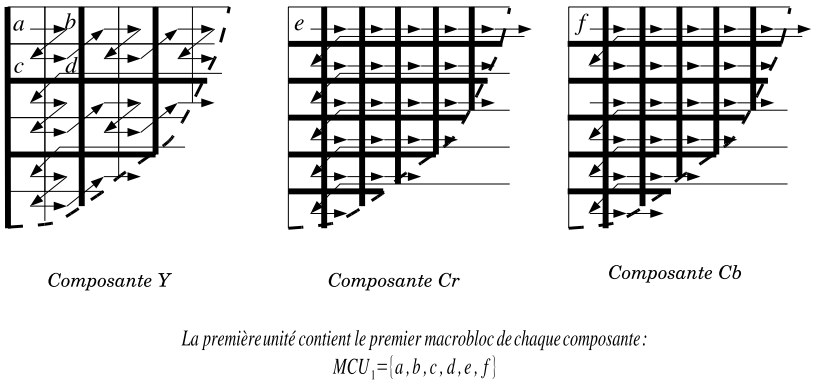


Figure 3.4. *Le MCU et l'ordre d'acquisition des images couleurs*

Une fois l'image partitionnée en MCU[2], celles-ci sont codées suivant l'ordre dans lequel elles sont construites. Pour chaque MCU, le traitement porte sur ses blocs indépendamment les uns des autres. Ainsi, pour une image couleurs, les quatre blocs de luminance sont codés avant les deux blocs de chrominance.

3.1.2. *La transformation*

Chaque bloc est ensuite analysé en termes de fréquences. La transformée utilisée est la DCT; directe pour le codeur et inverse pour le décodeur. L'intérêt de découper l'image en blocs est de fournir une description spectrale relativement localisée (voir chapitre 2 volume 1).

Mais, avant tout, afin de permettre une manipulation des réels sans risque de débordement, les valeurs sont centrées en zéro. Si, par exemple, l'intervalle des valeurs est initialement compris entre 0 et 255, ces valeurs sont translatées pour être comprises entre -128 et $+127$. Dans l'en-tête du flux de compression est enregistré la constante P indiquant la profondeur des pixels. Il suffit au décodeur de récupérer cette valeur et d'ajouter 2^{P-1} aux valeurs reconstruites pour annuler le centrage des valeurs.

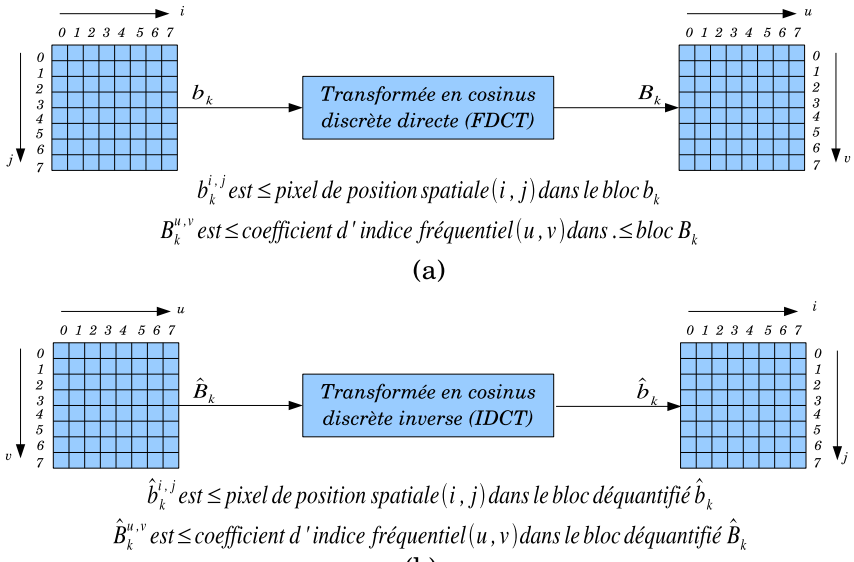


Figure 3.5. *Analyse spectrale d'un bloc : (a) transformée directe du codeur et (b) transformée inverse du décodeur*

2. Dans le cas des images à niveaux de gris, la MCU contient un seul bloc.

Une fois le centrage fait, la DCT est appliquée sur les blocs suivant l'ordre dans lequel ils sont enregistrés dans les MCU. Pour chaque bloc, le résultat de la transformée est un bloc de taille 8×8 dont les valeurs sont les coefficients des fréquences.

La figure 3.5 montre le schéma général de la transformation directe utilisée par le codeur et de la transformation inverse utilisée par le décodeur.

Le chapitre 2 du volume 1 fournit des exemples d'application des transformées en cosinus directe et inverse.

3.1.3. *La quantification*

Le coefficient d'indices $(0, 0)$, appelé coefficient DC, correspond à la moyenne d'intensité du bloc. Les coefficients suivants, appelés coefficients AC, s'ajoutent à cette moyenne. Ils apportent une information de plus en plus fine et de plus en plus localisée au fur et à mesure que les indices augmentent.

Le chapitre 2 du volume 1 précise que les fréquences augmentent avec les indices. De plus, quand les fréquences augmentent, les amplitudes de leurs coefficients diminuent et l'information sous-jacente est mieux localisée. Ainsi, plus la fréquence est élevée, moins elle est perceptible.

De ce fait, la quantification utilisée est une *quantification matricielle* d'équation :

$$Q_k^{u,v} = \mathrm{round}\left(\frac{B_k^{u,v}}{S^{u,v}}\right)$$

où $\mathrm{round}(x)$ fournit la valeur entière la plus proche de x, et, où $S^{u,v}$ est la pondération du coefficient d'indice (u, v). L'ensemble des pondérations est regroupé en une *matrice de quantification*.

Aucune spécification ne vient préciser les valeurs à utiliser pour cette matrice; le choix est laissé au développeur. Toutefois, le standard montre la forme que celle-ci peut prendre. Il propose de distinguer la quantification de la luminance, de la quantification de la chrominance.

Le tableau 3.1 montre les matrices proposées pour des valeurs entre -128 et $+127$. Dans les deux cas, les pondérations augmentent avec les indices. Ainsi, – et en accord avec les caractéristiques du système visuel humain – les amplitudes vont en diminuant avec l'augmentation des fréquences. La chrominance est plus fortement pondérée que la luminance.

Le standard précise que ces matrices produisent des détériorations visibles, mais, qu'avec des pondérations de moitié, les détériorations sont imperceptibles.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
72	92	95	98	112	100	103	99

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Tableau 3.1. *A gauche : la matrice de quantification de la luminance Y ; à droite : la matrice de quantification des axes Cr et Cb, de la chrominance*

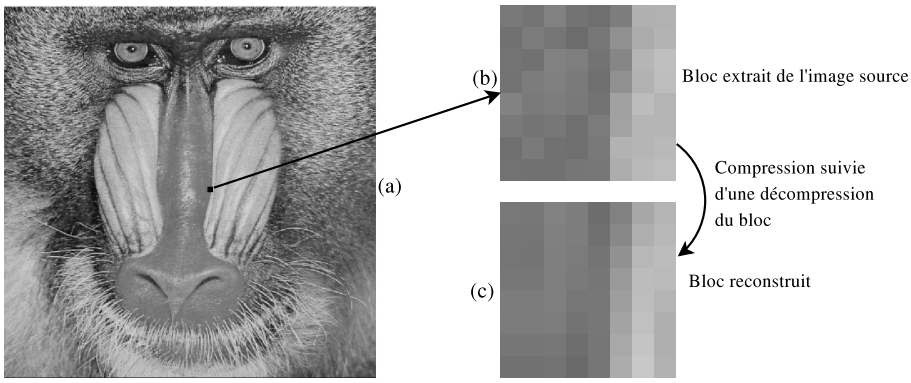


Figure 3.6. *(a) Une photographie d'un mandrille où en noir apparaît le bloc* 8×8 *utilisé ; (b) le bloc initial ; (c) le bloc reconstruit*

EXEMPLE 3.2.– *La figure 3.6 illustre l'effet de la quantification sur un bloc. Pour visualiser l'effet sur l'ensemble de l'image, reportez à la section Transformée en consinus discrète du chapitre 1 du volume 1. La figure 3.6a est la photographie d'un babouin d'où est extrait le bloc : sa position est indiquée en noir à peu près au centre. La figure 3.6b présente un agrandissement de ce bloc. La figure 3.6c montre ce bloc une fois reconstruit; c'est-à-dire après application de la transformation directe, de la quantification puis de la déquantification et de la transformation inverse. La matrice de quantification est celle du tableau 3.1. Les pertes sont ici perceptibles ; on voit apparaître des zones bien plus uniformes dans le bloc reconstruit.*

Les pertes sont encore plus visibles à la lecture des valeurs numériques. Le tableau 3.2a montre les valeurs du bloc original. Le tableau 3.2b montre les valeurs du bloc fréquentiel obtenu grâce à la DCT direct. Les coefficients en première ligne sont nettement plus forts que les autres, du fait qu'il y ait deux régions verticales – une sombre et une claire – dans le bloc original. Ensuite, le tableau 3.2c montre ce même bloc fréquentiel une fois quantifié avec les valeurs du tableau 3.1. On constate

que les pertes ont bien lieu à ce moment de la compression ; nombre de coefficients sont passés à zéro. Puis, le tableau 3.2d montre le bloc fréquentiel déquantifiée. Enfin, la transformée inverse est appliquée au bloc fréquentiel (voir tableau 3.2e). Du fait de l'annulation de certaines hautes fréquences, le bloc apparaît maintenant plus homogène en intensité, moins contrasté.

(a)

120.60000	118.50000	129.60000	114.50000	112.50000	127.70000	173.10000	179.60000
115.10000	120.60000	118.10000	110.80000	118.00000	143.90000	170.40000	181.60000
116.30000	116.80000	129.50000	126.50000	115.50000	137.10000	179.50000	190.60000
117.50000	121.50000	127.80000	122.10000	112.10000	143.20000	179.00000	191.00000
127.00000	125.50000	126.40000	118.00000	115.80000	159.50000	187.80000	180.60000
121.60000	120.40000	116.40000	113.70000	124.50000	163.10000	182.50000	181.20000
113.40000	117.20000	115.80000	113.2000	120.60000	174.70000	180.60000	186.20000
114.60000	114.90000	115.20000	115.00000	120.50000	174.70000	186.70000	188.9 0000

(b)

1112.10000	-187.95374	103.24351	-18.67685	-37.32500	22.25405	5.96985	-5.88592
-17.83132	24.70719	5.86313	-30.32555	13.05862	3.98614	-9.50977	14.68700
-10.23977	-6.02314	-2.67966	6.17588	0.19864	-5.84580	3.35475	0.62962
-1.90210	9.60292	1.46187	10.80774	-4.01659	-1.44424	-1.34605	4.51365
3.25000	5.04182	5.17593	-5.80870	-2.87500	7.20880	-0.43917	2.07749
-2.13315	-0.12229	-0.24921	-11.48347	3.48182	1.83133	1.45325	0.53445
2.45551	1.67988	-0.42025	-5.90762	5.24850	6.40379	-3.24534	5.20169
5.23885	1.49777	-0.51829	-1.91767	-2.62944	6.13310	0.99441	0.55374

(c)

0	-17	10	-1	-2	1	0	0
-1	2	0	-2	1	0	0	0
-1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(d)

285600	-47685	25500	-4080	-12240	10200	0	0
-3060	6120	0	-9690	6630	0	0	0
-3570	0	0	0	0	0	0	0
0	4335	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(e)

121.11784	119.66944	130.51912	125.82437	109.79322	130.20112	168.11061	179.82790
118.10940	117.36262	128.54868	124.98021	112.57709	136.85641	175.10228	184.77605
116.60146	116.44596	126.95070	123.76101	116.21903	145.94948	183.51402	188.70673
119.54725	119.24302	127.15524	122.23628	118.63826	153.26679	188.23881	186.65544
123.99913	123.12087	127.46375	119.67585	119.26739	158.20037	189.60305	180.27227
124.26613	123.23914	124.98950	115.47234	119.00784	162.83889	191.64905	175.54220
118.96982	118.53411	119.60314	110.46477	118.86141	168.14359	196.27242	175.68450
113.28260	113.54851	114.95392	106.94182	118.96649	172.12009	200.58531	177.95386

Tableau 3.2. *(a) Le bloc original où l'on voit apparaître à droite une zone plus claire ; (b) le bloc fréquentiel correspondant dont les coefficients de la première ligne sont les plus élevés ; (c) le même bloc fréquentiel quantifié ; (d) le bloc fréquentiel déquantifié ; (e) le bloc reconstruit grâce à la transformation inverse.*

3.1.4. *Le codage*

Au sein d'un bloc, le codage du coefficient DC diffère du codage des coefficients AC. Le coefficient DC est codé en mode prédictif suivant le principe du DPCM (voir figure 3.7). La prédiction correspond au coefficient DC du bloc précédemment codé; lors du codage du premier coefficient DC la prédiction est nulle.

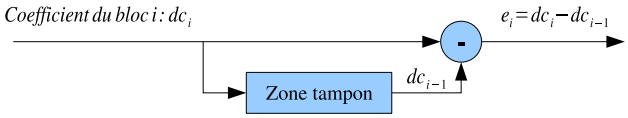


Figure 3.7. *Codage prédictif des coefficients DC*

Ensuite, dans un souci d'optimisation, les coefficients AC quantifiés du bloc B_k sont rangés dans une liste suivant un *parcours en zigzag*. Cette technique suit l'augmentation des indices dans le bloc (voir figure 3.8) qui correspond à l'augmentation en fréquences. Ceci se justifie par le fait que les hautes fréquences sont plus atténuées que les basses fréquences. Lorsque tous les coefficients non nuls sont rangés dans la liste le parcours s'arrête et le marqueur EOB[3] vient cloturer la liste. Le tableau 3.3 montre le résultat du parcours en zigzag pour l'exemple illustré par la figure 3.6 et le tableau 3.2c.

-17	-1	-1	2	10	-1	0	0	0	0	1	0	-2	-2	1	1	EOB

Tableau 3.3. *Liste obtenue du parcours en zigzag du bloc en tableau 3.2c*

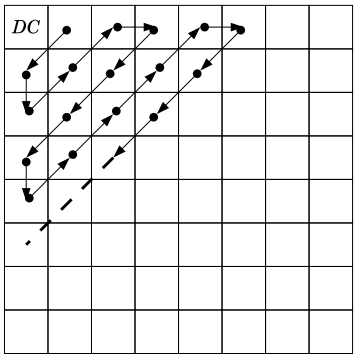


Figure 3.8. *Principe du parcours en zigzag : le coefficient DC n'est pas pris en compte*

3. EOB : *End Of Bloc*.

Ensuite, un algorithme de type RLC (voir section *Codage par plages* du chapitre 3 du volume 1) est appliqué pour les valeurs nulles. Pour l'exemple du tableau 3.2.c, nous obtenons :

$$(0,17)(0,-1)(0,-1)(0,2)(0,10)(0,-1)(4,1)(1,-2)(0,-2)(0,1)(0,1)(0,0)$$

où $(0,0)$ est la marque de fin de liste EOB.

Pour terminer, un codage entropique transforme cette liste et l'erreur de prédiction du coefficient DC en un flux binaire. Ce codage entropique est, par défaut, une variante de l'algorithme de Huffman. Mais, le codeur arithmétique binaire QM, peut être utilisé (voir annexe B du volume 1). L'algorithme de Huffman utilise jusqu'à quatre tables de codage.

3.2. Le mode progressif

Ce mode est une extension du mode séquentiel.

Il utilise le fait que le flux binaire code les coefficients fréquentiels des blocs. Ce mode est dit progressif puisqu'un affichage, à l'aide des seules basses fréquences, peut se faire pendant que les hautes fréquences sont décodées. Dans ce cas, l'affichage est une approximation de l'image initiale.

Il existe deux possibilités : la *sélection fréquentielle* et l'*approximation successive*. Mais, toute implémentation du codeur ou du décodeur doit permettre l'utilisation conjointe des deux options : la sélection fréquentielle vient compléter l'approximation successive.

La figure 3.9 présente un comparatif schématique entre le mode séquentiel et les options du mode progressif :

– la sous-figure (a) décrit l'organisation des blocs. Le parcours en zigzag donne une suite d'indices de coefficients allant de 0 à 63; 0 est l'indice du coefficent DC et 63 celui des plus hautes fréquences. Cette suite est représentée verticalement sur le graphique, alors qu'horizontalement, sont schématisés les bits avec l'indice 7 pour le bit de poids fort et 0 pour le bit de poids faible. La profondeur de codage des coefficients peut être de 8 bits, comme dans la figure, ou de 12 bits. Les blocs sont rangés les uns derrière les autres (dans le sens de la profondeur sur le graphique) ;

– la sous-figure (b) décrit le *mode séquentiel*. L'ensemble des bits de tous les coefficients du premier bloc sont transférés au codeur entropique, puis ceux du deuxième bloc et ainsi de suite suivant l'ordre imposé par la construction des MCU ;

– la sous-figure (c) décrit le *mode progressif à sélection fréquentielle*. Ce mode opère en plusieurs passes. La première passe transfert les coefficients DC de tous les blocs. Ceci permet au décodeur de construire une première approximation de l'image.

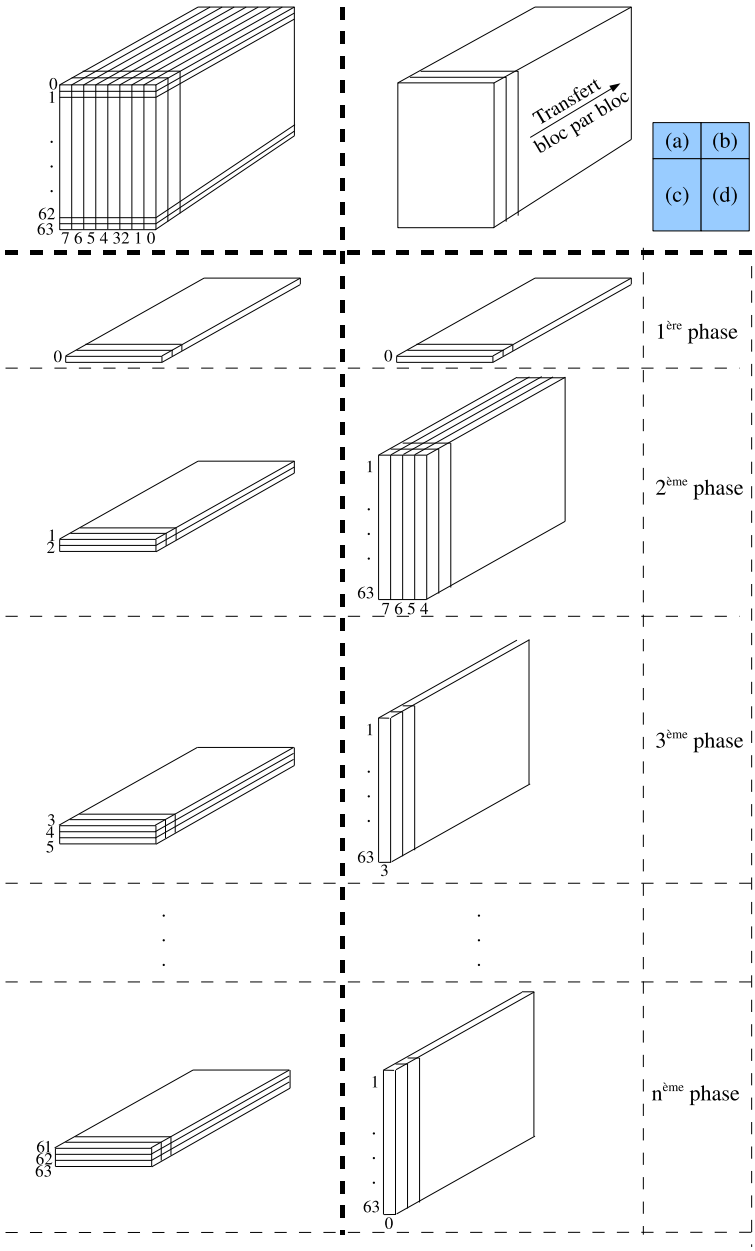


Figure 3.9. *L'ordre de lecture et d'envoie des coefficients des blocs selon le mode : (a) l'organisation des coefficients fréquentiels quantifiés, (b) le mode séquentiel, (c) le mode progressif à sélection fréquentielle, (d) le mode progressif à approximation successive (d'après W.B. Pennebaker [PEN 93]).*

Puis, la deuxième passe vient affiner cette approximation; les coefficients d'indices 1 et 2 sont alors transférés pour l'ensemble des blocs. Le procédé est répété jusqu'à ce que les coefficients d'indice 63 soient transférés. Le nombre de passes est fixé par le partitionnement de l'ensemble des indices $\{0, \cdots, 63\}$. Ce partitionnement est laissé à la discrétion du développeur. Aussi, l'en-tête du flux binaire doit contenir ce partitionnement pour que le décodeur puisse faire les mêmes passes ;

– la figure (d) décrit le *mode progressif à approximation successive*. Il fonc-tionne également en plusieurs passes. Comme pour l'option de sélection fréquentielle, les coefficients DC sont transférés en premier. Puis, les coefficients sont divisés par une puissance de 2 (décalage à droite des bits); la puissance est précisée dans l'entête du flux binaire. Le décodeur effectue l'opération inverse : multiplication par la même puissance de 2. Une fois les premiers transferts faits, les valeurs des coefficients sont complétées, bit après bit, à chaque transfert.

3.3. Le mode hiérarchique

L'objectif du mode hiérarchique est similaire à celui du mode progressif : un affichage approximatif de l'image peut être fait avant que l'ensemble du flux binaire soit décodé. Mais, au lieu de travailler sur la profondeur des coefficients (progressif avec approximations successives) ou de les regrouper en paquets (progressif avec sélection fréquentielle), une image basse résolution est construite à partir de l'image originale. Plusieurs étapes de *sous-échantillonnage* sont appliquées. Pour ce faire, chaque sous-échantillonnage divise la longueur et la largeur de l'image par deux. Le procédé est expliqué plus en détail en fin de section.

Cette image basse résolution est alors codée en mode séquentiel, progressif ou sans perte. Elle est transférée au codeur entropique qui l'enregistre dans le flux binaire. Le décodeur reçoit donc cette image basse résolution qu'il peut décoder et afficher comme première approximation.

Ensuite, l'image basse résolution est *suréchantillonné* suivant un procédé inverse au sous-échantillonnage précédent (ce procédé est décrit ci-après). Le résultat est une image qui a doublé sa largeur et sa longueur. Cette nouvelle image n'est pas forcément identique à la version sous-échantillonnée de l'image originale de taille identique. En effet, la boucle sous-échantillonnage/suréchantillonnage n'est pas tenue de fournir une reconstruction exacte. Aussi, ce n'est pas l'image suréchantillonnée qui est transférée au codeur entropique, mais, sa différence avec la version sous-échantillonnée de l'image originale. Le procédé est similaire aux techniques de prédiction (DPCM).

Le processus est répété jusqu'à ce que l'image suréchantillonnée ait atteint la taille de l'image originale. Ceci signifie que si n sous-échantillonnages successifs ont été appliqués à l'image originale pour obtenir l'image basse résolution, n suréchantillonnages de l'image basse résolution sont appliqués. Chaque suréchantillonnage est suivi

d'un codage entropique de la différence entre l'image reconstruite à partir de l'image basse et l'image de même résolution provenant du sous-échantillonnage de l'image originale.

Le principe du sous-échantillonneur n'est pas précisé par le standard ; il est laissé à la discrétion des développeurs. En revanche, il doit être compatible avec le suréchantillonneur spécifié par le standard. Ce dernier correspond à l'équation suivante :

$$\tilde{p} = \lfloor (\hat{a} + \hat{b})/2 \rfloor \tag{3.1}$$

où $\hat{a}$ et $\hat{b}$ sont des pixels adjacents de l'image à suréchantillonner.

Le (sous-)suréchantillonnage peut- être appliqué soit en lignes, soit en colonnes, soit sur les deux à la fois. Dans le dernier cas, le (sous-)suréchantillonnage est d'abord fait en lignes puis en colonnes sur le résultat du (sous-) suréchantillonnage en lignes.

REMARQUE 3.1.– *Le sous-échantillonnage correspond à une convolution avec un filtre passe-bas.*

EXEMPLE 3.3.– *Soit un signal monodimensionnel, x_0, de la forme :*

$$x_0 = \boxed{5}\boxed{15}\boxed{35}\boxed{10}\boxed{15}\boxed{15}\boxed{20}\boxed{20}$$

Le sous-échantillonnage choisi opère en deux passes. D'abord, il effectue la convolution du signal suivant l'équation[4] *:*

$$\tilde{p} = \left\lfloor \frac{a}{4} + \frac{p}{2} + \frac{b}{4} \right\rfloor$$

où a et b sont les pixels voisins de p. Dans cette écriture, il faut comprendre que p est remplacé par sa prédiction $\tilde{p}$. Avec le signal source, on obtient :

$$\boxed{7}\boxed{17}\boxed{23}\boxed{17}\boxed{13}\boxed{16}\boxed{18}\boxed{20}$$

La deuxième étape construit le signal d'approximation x_1. Pour ce faire, elle ne retient que les échantillons pairs du signal convolué :

$$x_1 = \boxed{7}\boxed{23}\boxed{13}\boxed{18}$$

En effectuant à nouveau un sous-échantillonnage, on obtient l'approximation :

$$x_2 = \boxed{11}\boxed{16}$$

Celle-ci est envoyée au décodeur – via *le codeur entropique. Puis, le codeur envoie l'erreur de prédiction $\hat{e}_1$, qui a une entropie plus faible que x_1. L'erreur $\hat{e}_1$ est obtenue en deux passes. Premièrement, le signal d'approximation x_2, est suréchantillonné*

4. Pour opérer correctement la convolution, les valeurs extrèmes, gauche et droite, sont dupliquées à gauche et à droite du signal respectivement.

(voir équation (3.1)); on obtient le signal de prédiction[5] *:*

$$\hat{x}_1 = \boxed{11}\boxed{13}\boxed{16}\boxed{16}$$

Puis, l'erreur de prédiction $\hat{e}_1 = x_1 - \hat{x}_1$ *est calculée et transférée :*

$$\hat{e}_1 = \boxed{-4}\boxed{+10}\boxed{-3}\boxed{+2}$$

Le procédé est répété pour le signal x_0*. L'erreur de prédiction est envoyée au décodeur :*

$$\hat{e}_0 = \boxed{-2}\boxed{+0}\boxed{+12}\boxed{-8}\boxed{+2}\boxed{+0}\boxed{+2}\boxed{+2}$$

Le décodeur reçoit donc en premier l'approximation x_2*, qu'il peut afficher. Puis, il reçoit le signal d'erreur de prédiction* $\hat{e}_1$*. Il peut alors reconstruire le signal d'approximation* $x_1 = \hat{x}_1 + \hat{e}_1$*. Quand il reçoit le signal d'erreur de prédiction* $\hat{e}_0$*, le signal original* x_0 *peut être affiché.*

3.4. Synthèse

Conceptuellement, le standard JPEG propose plusieurs modes de compression dont un sans perte. Ce dernier utilisant des outils spécifiques, sa présentation est faite dans le précédent chapitre (voir section 2.3).

La compression avec perte opère une analyse fréquentielle locale de l'image. Les pertes engendrées sont contrôlées de manière à ce qu'elles restent invisibles à l'utilisateur ou, à défaut, peu génantes dans l'utilisation qu'il fait de l'image.

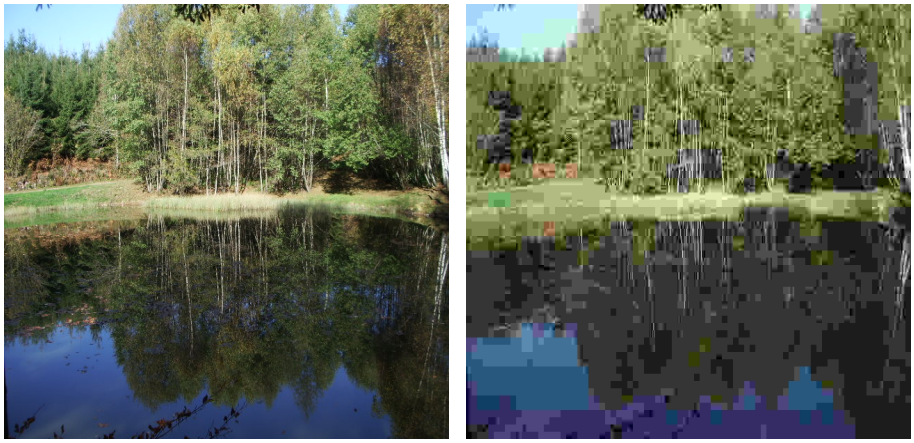
L'analyse fréquentielle locale divise l'image en bloc de taille fixe (8×8). Chaque bloc est, ensuite, transformé par une DCT directe. Les valeurs du bloc fréquentiel résultant fournissent une mesure indiquant l'importance des fréquences dans le bloc original.

Les hautes fréquences étant locales et d'amplitudes plus faibles que les basses fréquences, leurs dégradations, voire disparitions, n'entraînent pas des déformations trop perturbantes pour l'utilisateur.

Ainsi, JPEG utilise une quantification matricielle appliquée à chaque bloc qui diminue plus fortement les hautes fréquences que les basses fréquences. Mais, il existe une limite à la compression qui, lorsqu'elle est dépassée, engendre des artéfacts.

En effet, la compression se faisant indépendamment sur chaque bloc, les contours des blocs apparaissent lors de l'affichage de l'image décompressée, si la quantification appliquée par le codeur est trop forte. La figure 3.10 illustre ce phénomène.

5. Le dernier échantillon est recopié à sa droite pour que la prédiction est la taille adéquate.



(a) (b)

Figure 3.10. *Effet de distorsion à forte compression : (a) image faiblement compressée (150 Ko) ; (b) image fortement compressée (7,6 Ko)*

Le chapitre suivant traite du standard JPEG2000 qui présente des taux de compression bien plus élevés à qualité égale avec des artéfacts moins génants. Il reste cependant d'utilisation marginale, compte tenu de la forte implantation industrielle des codeurs et décodeurs JPEG et de l'utilisation de son principe (découpage en blocs et DCT) par les codeurs vidéo MPEG (voir chapitres 5, 6 et 7).

Chapitre 4

JPEG2000

Ce chapitre détaille la première partie (*Part-1*) du format JPEG2000. Les parties suivantes (*Part-2*, etc.) sont des extensions et des améliorations de la première partie.

La différence majeure avec les formats étudiés, dans les précédents chapitres, est l'utilisation de la transformée en ondelettes (section 4.2) qui fournit une description *locale* des fréquences spatiales (voir chapitre 2 volume 1). C'est une représentation *multirésolution* comme l'illustre la figure 4.1. Chaque niveau de résolution fournit quatre sous-images : une sous-image d'approximation LL et trois sous-images LH, HL et HH, respectivement, pour les détails horizontaux, verticaux et diagonaux.

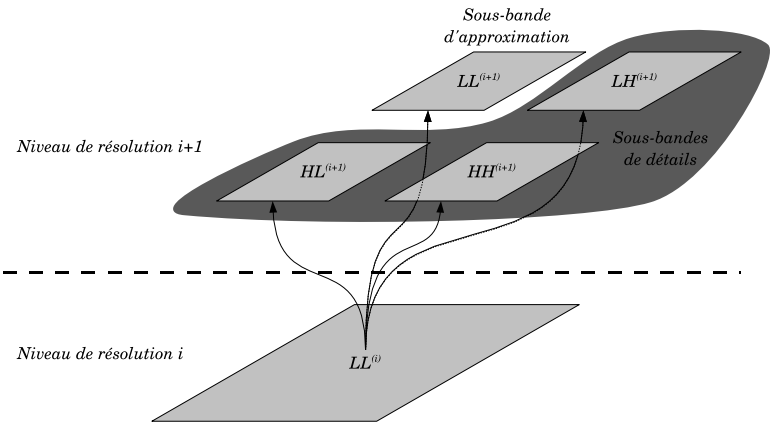


Figure 4.1. *Décomposition de la sous-bande d'approximation* $\mathrm{LL}^{(i)}$*, de niveau* i *de résolution, en quatre sous-bandes de niveau* $i+1$*: une d'approximation* $\mathrm{LL}^{(i+1)}$*, et trois de détails* $\mathrm{HL}^{(i+1)}$*,* $\mathrm{LH}^{(i+1)}$ *et* $\mathrm{HH}^{(i+1)}$

Ces quatre sous-images permettent la reconstruction de la sous-image d'approximation du niveau précédent. L'image originale est alors considérée comme l'image d'approximation initiale, c'est-à-dire celle de plus haute résolution. Chaque composante de l'image originale est décrite par des coefficients à différents niveaux de résolution, résultant de l'analyse par transformée en ondelettes.

DÉFINITION 4.1.– *Les sous-images sont parfois appelées* sous-bandes *en référence aux filtres d'analyse et de synthèse de la transformée en ondelettes. Cette terminologie est reprise, ici, afin de pouvoir utiliser le terme de sous-image pour une autre définition.*
Si une image est de taille trop importante, elle peut être découpée en sous-images *(ou tuiles). Cependant cette technique n'est habituellement pas utilisée par le codeur JPEG2000 pour la gestion des images volumineuses. On préfère séparer l'image en composantes.*
Sachant que les composantes *sont indépendantes*[1]*, chacune d'elles possède sa propre dynamique et ses propres dimensions. Chaque composante est donc traitée indépendamment des autres composantes.*
Si une composante reste volumineuse, elle est alors découpée en pavés. Les pavés *(*precinct*) sont constitués d'un nombre entier de codes-blocs et respectent certaines propriétés liées aux flux binaires finaux (voir section 4.5).*
Les codes-blocs *sont les unités servant au codage. Ils représentent une région de taille* (32×32) *ou* (64×64) *de la composante.*
En résumé, le codage au format JPEG2000 s'effectue sur les codes-blocs de chaque pavé, de chaque sous-bande et de chacune des composantes de l'image originale.

La figure (4.2) montre le schéma général de fonctionnement de la première partie (*Part-1*) du standard. Tout d'abord, le codeur effectue un certain nombre de prétraitements. Le premier d'entre eux consiste à formater l'image afin d'accéder facilement à n'importe quelle position de n'importe quelle composante de l'image.

Ensuite, le deuxième prétraitement consiste à ramener à zéro la moyenne des échantillons de la composante, afin de faciliter la quantification.

Le dernier des prétraitements concerne les images couleurs. Celles-ci sont parfois au format RGB qui présente une forte corrélation entre les trois composantes. Afin de décorréler ces composantes et de s'adapter au système visuel humain, le modèle de couleurs YUV est choisi. Comme expliqué dans le chapitre 4 du volume 1, ce modèle permet de séparer l'information luminosité (Y) de la chrominance (U et V).

La section suivante (4.1) présente l'ensemble de ces prétraitements.

1. Par exemple, les trois composantes indépendantes peuvent être les trois canaux couleurs après prétraitement ou encore une composante texte, une composante logo et une photographie.

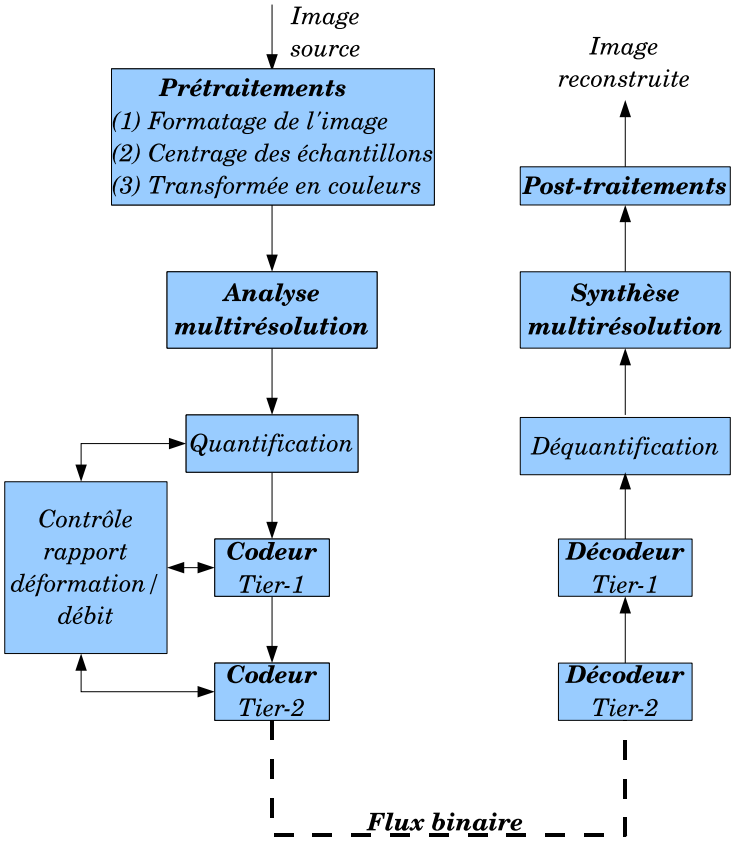


Figure 4.2. *Fonctionnement général de la partie 1 du standard JPEG2000 : à gauche le codeur ; à droite le décodeur*

Une fois les prétraitements effectués, une analyse en ondelettes, expliquée en section 4.2, est effectuée. Elle est suivie d'une quantification scalaire à zone morte décrite en section 4.3. Puis, le codage binaire est effectué.

La décompression suit tout simplement le processus inverse.

JPEG2000 autorise deux modes de compression : un mode avec pertes et un autre sans perte. Le mode sans perte est dit *réversible*, alors que celui avec pertes est dit *irréversible*. Le mode est choisi avant de commencer la compression et influence :

– le centrage des échantillons ;

– la transformée en couleur ;

– la transformée en ondelettes.

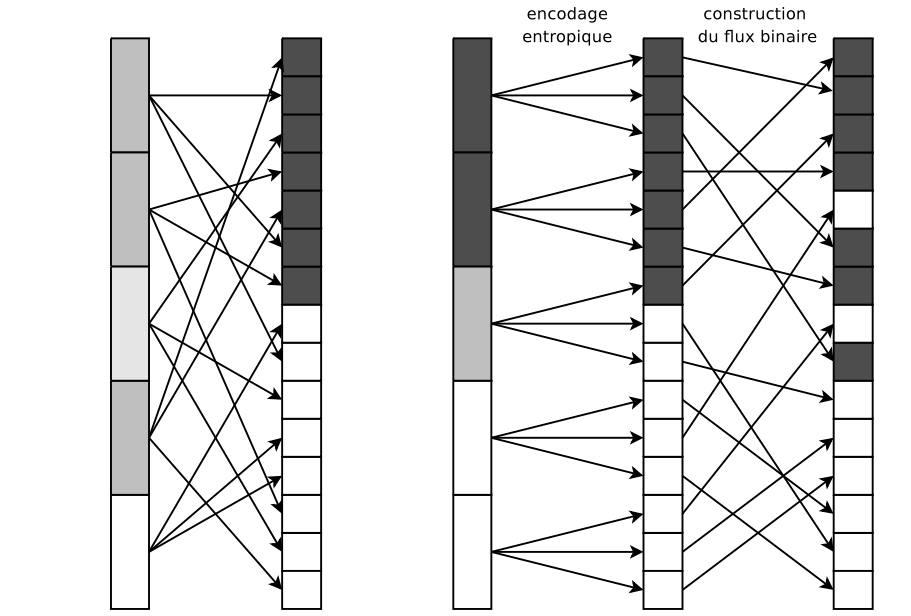


Figure 4.3. *Codage (a) direct, (b) en deux phases. Dans les deux cas, la colonne de gauche représente l'information à coder. Les algorithmes de codage direct n'offre aucun contrôle sur l'ordre dans lequel l'information finale va être délivrée. En revanche, avec un codage en deux phases, la première phase (passage de la colonne de gauche à la colonne intermédiaire) effectue le codage, puis, la deuxième phase (passage de la colonne intermédiaire à celle de droite) ordonne l'information à délivrer suivant des critères fournis par l'utilisateur ou le matériel.*

Quel que soit le mode choisi, les algorithmes sont identiques; seuls les paramètres diffèrent. En mode réversible, les paramètres sont des *valeurs entières exactes*, alors que le mode irréversible fournit des *valeurs réelles* approchant les paramètres exacts. Evidemment, la compression avec pertes offre des taux de compression supérieurs à ceux de la compression sans perte.

Le codage binaire opère en deux phases, *Tier-1* (section 4.4) et *Tier-2* (section 4.5), sur lesquelles vient se greffer un module de contrôle de qualité jouant sur le rapport entre la déformation de l'information (les pertes d'information) et le débit souhaité ou autorisé. Ce module est laissé à la discrétion de l'utilisateur [ACH 04] et permet, par exemple :

– d'éviter des délais d'attente trop longs à la réception de l'image. Ces délais sont généralement provoqués par des débits faibles, variables et pouvant être soumis à de

fortes pertubations. Dans ce cas, la qualité de la résolution de l'image au décodage est quel que peu négligée au profit d'un transport plus sûr. C'est le cas, par exemple, d'un affichage sur un mobile ;

– d'optimiser le rendu à la réception pour un travail professionnel exploitant les informations des images. Dans ce cas, le réseau doit offrir un débit suffisant et une bonne imperméabilité aux perturbations.

REMARQUE 4.1.– *Les valeurs optimales du rapport déformation/débit sont déjà prises en compte au sein des deux phases principales, Tier-1 et Tier-2, détaillées en sections 4.4 et 4.5.*

La phase *Tier-1* effectue un codage *entropique* de la composante en la divisant en blocs codés indépendamment les uns des autres. Puis, la phase *Tier-2* organise les flux binaires de la phase *Tier-1* en paquets optimums suivant la résolution et la qualité recherchées pour l'image reconstruite.

La séparation en deux phases du codage permet de mieux prendre en compte les contraintes fournies par le module de contrôle. La figure 4.3 illustre ce phénomène.

Lors de la compression, il est également possible de définir des régions d'intérêt (ROI[2]) de manière interactive. L'utilisateur sélectionne les régions contenant une information pertinente. Puis, grâce au principe de codage en deux phases, un simple décalage de bits permet de les coder. Ces régions sont alors compressées moins fortement que le reste de l'image et décompressées plus rapidement.

4.1. Les prétraitements

Il existe trois prétraitements optionnels au sein du standard JPEG2000 :

– structuration des images ;

– centrage des échantillons ;

– choix de l'espace multicomposantes.

4.1.1. *Structuration*

Comme pour JPEG, une image peut être constituée de textes, de graphiques, de photographies, etc. Chaque texte, graphique ou photographie est alors une *composante* de l'image. Les composantes peuvent également correspondre aux couleurs d'une image.

2. ROI : sigle anglais pour *Regions Of Interest*.

Pour gérer ces différentes configurations, chaque composante dispose alors de ses propres dimensions.

JPEG2000 gère également les images de grandes tailles comme celles fournies par les satellites. Ces composantes de grandes dimensions ne peuvent être manipulées dans leur ensemble. Le standard partitionne chaque composante en *sous-images* (*tiles*) et en *pavés* (*precincts*).

Il faut donc définir une structure – appelée *canevas* – permettant une manipulation aisée des composantes, des sous-images, des pavés et des sous-bandes; c'est-à-dire une manipulation à l'aide d'un seul jeu de coordonnées.

L'origine du canevas est $(0, 0)$ et ses indices sont positifs. La position de chaque pixel y est définie de manière unique :

$$\mathbf{n} = (n_1, n_2) \in [I_1, S_1[\times[I_2, S_2[$$

avec I_i (resp. S_i) la borne inférieure (resp. supérieure) des lignes, quand $i = 1$, et des colonnes, quand $i = 2$, de l'image au sein du canevas.

De même, chaque composante $\boldsymbol{c}$ y est définie par ses coordonnées et ses facteurs $(F_1^{(\boldsymbol{c})}, F_2^{(\boldsymbol{c})})$ de sous-échantillonnage[3]. Ainsi, la position du pixel x au sein de la composante $\boldsymbol{c}$ est :

$$x^{(\boldsymbol{c})}[\boldsymbol{n}] = x^{(\boldsymbol{c})}[n_1, n_2] = (n_1 F_1^{(\boldsymbol{c})}, n_2 F_2^{(\boldsymbol{c})})$$

Autrement dit :

$$(n_1, n_2) \in \left[I_1^{(\boldsymbol{c})}, S_1^{(\boldsymbol{c})}\right[\times \left[I_2^{(\boldsymbol{c})}, S_2^{(\boldsymbol{c})}\right[\Leftrightarrow (n_1 F_1^{(\boldsymbol{c})}, n_2 F_2^{(\boldsymbol{c})}) \in [I_1, S_1[\times[I_2, S_2[$$

et les bornes d'une composante s'écrivent :

$$I_i^{(\boldsymbol{c})} = \left\lceil \frac{I_i}{F_i^{(\boldsymbol{c})}} \right\rceil \text{ et } F_i^{(\boldsymbol{c})} = \left\lceil \frac{S_i}{F_i^{(\boldsymbol{c})}} \right\rceil, i = 1, 2$$

Comme le montrent les équations ci-dessus, il est possible de désigner sans ambiguïté une position au sein d'une composante à l'aide d'un seul jeu de coordonnées, et de préciser de quel objet il s'agit en lui assignant un exposant parenthèsé.

Par exemple, $I_2^{(\mathbf{s},\boldsymbol{c},r)}$ désigne la borne inférieure des colonnes au sein de la sous-image $\mathbf{s}$, de la composante $\boldsymbol{c}$, au niveau de résolution r.

3. Il s'agit du facteur de redimensionnement par rapport à la taille originale de l'image.

Les facteurs de sous-échantillonnage peuvent varier de 1 à 255. Mais, dans la plupart des applications, ces facteurs prennent des valeurs entières dans l'ensemble $\{1, 2, 4\}$.

Munie d'un canevas, l'image peut maintenant être découpée en sous-images et pavés.

4.1.1.1. *Les sous-images*

Une partition $\boldsymbol{S}$ en sous-images est définie par son origine sur le canevas :

$$\left(O_1^{(\boldsymbol{S})}, O_2^{(\boldsymbol{S})}\right)$$

et par la taille de ses sous-images :

$$\left(T_1^{(\boldsymbol{S})}, T_2^{(\boldsymbol{S})}\right)$$

Les indices $\boldsymbol{s} = (s_1, s_2)$ des sous-images s'inscrivent alors dans l'intervalle :

$$\boldsymbol{s} \in [O_1^{(\boldsymbol{S})}, O_1^{(\boldsymbol{S})} + T_1^{(\boldsymbol{S})} N_1^{(\boldsymbol{S})}] \times [O_2^{(\boldsymbol{S})}, O_2^{(\boldsymbol{S})} + T_2^{(\boldsymbol{S})} N_2^{(\boldsymbol{S})}]$$

où $N_1^{(\boldsymbol{S})}$ et $N_2^{(\boldsymbol{S})}$ désignent le nombre de sous-images, respectivement, en colonne et en ligne :

$$N_i^{(\boldsymbol{S})} = \frac{S_i - O_i^{(\boldsymbol{S})}}{T_i^{(\boldsymbol{S})}}$$

De plus, la partition doit obligatoirement recouvrir l'ensemble de l'image :

$$\left\{ \begin{array}{ll} 0 \leq & O_i^{(\boldsymbol{S})} < I_i, \\ 0 \leq & I_i - O_i^{(\boldsymbol{S})} < T_i^{(\boldsymbol{S})} \end{array} \right. \quad i = 1, 2$$

4.1.1.1.1. Les sous-images au sein d'une composante $\boldsymbol{c}$

Les bornes inférieure et supérieure de la sous-image s s'écrivent :

$$I_i^{(\boldsymbol{c},\mathbf{s})} = \begin{cases} \left\lceil \dfrac{I_i}{F_i^{(\boldsymbol{c})}} \right\rceil & \textbf{si } s_i = 0 \\ \left\lceil \dfrac{O_i^{(\boldsymbol{S})} + s_i T_i^{(\boldsymbol{S})}}{F_i^{(\boldsymbol{c})}} \right\rceil & \textbf{si } s_i \neq 0 \end{cases}$$

$$S_i^{(\boldsymbol{c},\mathbf{s})} = \begin{cases} \left\lceil \dfrac{S_i}{F_i^{(\boldsymbol{c})}} \right\rceil & \textbf{si } s_i = N_i^{(\mathbf{s})} - 1 \\ \left\lceil \dfrac{S_i^{(\boldsymbol{S})} + (s_i + 1) T_i^{(\boldsymbol{S})}}{F_i^{(\boldsymbol{c})}} \right\rceil & \textbf{si } s_i \neq S_i^{(\mathbf{s})} - 1 \end{cases}$$

4.1.1.1.2. Les sous-images au niveau de résolution r

Les bornes s'écrivent :

$$I_i^{(\boldsymbol{c},\mathbf{s},r)} = \left\lceil \frac{I_i^{(\boldsymbol{c},\mathbf{s})}}{2^{\mathfrak{D}^{(\boldsymbol{c},\mathbf{s})}-r}} \right\rceil \text{ et } S_i^{(\mathbf{s},\boldsymbol{c},r)} = \left\lceil \frac{S_i^{(\boldsymbol{c},\mathbf{s})}}{2^{\mathfrak{D}^{(\boldsymbol{c},\mathbf{s})}-r}} \right\rceil$$

$\mathfrak{D}^{(\boldsymbol{c},\mathbf{s})}$ étant le niveau maximum de résolution pour la sous-image s de la composante $\boldsymbol{c}$.

De plus, les sous-bandes de la résolution r sont indicées par des couples de valeurs binaires $\boldsymbol{b} = [b_1, b_2]$ tel que $(0,0)$ représente la sous-bande $LL^{(r)}$, $(0,1)$ la sous-bande $HL^{(r)}$, $(1,0)$ la sous-bande $LH^{(r)}$ et $(1,1)$ la sous-bande $HH^{(r)}$.

Ainsi $x^{(\boldsymbol{c},\mathbf{s},r,\mathbf{b})}[\mathbf{n}]$ désigne l'échantillon $\mathbf{n}$ de la sous-bande indicée par $\boldsymbol{b}$, de niveau de résolution r, dans la sous-image indicée par s, provenant de la composante $\boldsymbol{c}$. Les bornes de la sous-bande $\boldsymbol{b}$ sont :

$$\begin{aligned} I_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{b})} &= \left\lceil \frac{I_i^{(\boldsymbol{c},\boldsymbol{s})} - 2^{r-1}b_i}{2^r} \right\rceil \\ & \qquad\qquad r \in \{1,2,\ldots,\mathfrak{D}^{(\boldsymbol{c},\mathbf{s})}\} \\ S_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{b})} &= \left\lceil \frac{S_i^{(\boldsymbol{c},\boldsymbol{s})} - 2^{r-1}b_i}{2^r} \right\rceil \end{aligned}$$

REMARQUE 4.2.– *Les sous-images ont le même désavantage que les imagettes définies dans le standard JPEG : la séparation de l'information en morceaux indépendants les uns des autres peut engendrer des artefacts fort désagréables.*

Heureusement, avec le standard JPEG2000, il est rare que l'on ait recours aux sous-images. En effet, les pavés et les codes-blocs du codage entropique embarqué sont des outils évitant ces artefacts.

4.1.1.2. *Les pavés*

Comme annoncé par la définition 4.1 (page 72) de l'introduction, les pavés (*precincts*) sont utilisés pour découper les sous-bandes des composantes, afin de limiter la taille des paquets lors de la construction des flux binaires finaux (voir section 4.5). Ils sont directement associés aux codes-blocs de l'algorithme de codage décrit en section 4.4.

Tout comme pour les codes-blocs, leurs tailles doivent être des puissances de 2. Ainsi, un pavé contient un nombre entier de codes-blocs.

Chaque partition $\boldsymbol{P}$ en pavés est connue par son origine :

$$\left(O_1^{(\boldsymbol{P})}, O_2^{(\boldsymbol{P})}\right)$$

où $O_i^{(\boldsymbol{P})}$ $(i = 1, 2)$ vaut 0[4], et par la taille de ses pavés :

$$\left(T_1^{(\boldsymbol{P},\boldsymbol{c},\boldsymbol{s},r)}, T_2^{(\boldsymbol{P},\boldsymbol{c},\boldsymbol{s},r)}\right).$$

Un pavé est indexé par :

$$\boldsymbol{p} = (p_1, p_2) \in \left[0, N_1^{(\boldsymbol{P},\boldsymbol{c},\boldsymbol{s},r)}\right[\times [0, N_2^{(\boldsymbol{P},\boldsymbol{c},\boldsymbol{s},r)}\left[\right.$$

avec :

$$N_i^{(\boldsymbol{P},\boldsymbol{c},\boldsymbol{s},r)} = \begin{cases} \left\lceil \dfrac{I_i^{(\boldsymbol{c},\boldsymbol{s},r)} - O_i^{(\boldsymbol{P})}}{T_i^{(\boldsymbol{P},\boldsymbol{c},\boldsymbol{s},r)}} \right\rceil - \left\lceil \dfrac{I_i^{(\boldsymbol{c},\boldsymbol{s},r)} - O_i^{(\boldsymbol{P})}}{T_i^{(\boldsymbol{P},\boldsymbol{c},\boldsymbol{s},r)}} \right\rceil & \textbf{si } S_i^{(\boldsymbol{c},\boldsymbol{s},r)} > I_i^{(\boldsymbol{c},\boldsymbol{s},r)} \\ 0 & \textbf{si } S_i^{(\boldsymbol{c},\boldsymbol{s},r)} = I_i^{(\boldsymbol{c},\boldsymbol{s},r)} \end{cases}$$

étant le nombre de pavés par ligne ($i = 2$) et par colonne ($i = 1$).

REMARQUE 4.3.– *La taille des pavés est fonction à la fois de la sous-image et de la résolution considérées*[5].

Ainsi, les bornes d'un pavé $\boldsymbol{p}$ de la sous-image $\boldsymbol{s}$ dans la composante $\boldsymbol{c}$ et de résolution r valent :

$$I_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{p})} = \begin{cases} I_i^{(\boldsymbol{c},\boldsymbol{s},r)} & \textbf{si } p_i = 0 \\ O_i^{(\boldsymbol{P})} + T_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{p})} \left(p_i + \left\lfloor \dfrac{I_i^{(\boldsymbol{c},\boldsymbol{s},r)} - O_i^{(\boldsymbol{P})}}{T_i^{(\boldsymbol{p},\boldsymbol{s},c,r)}} \right\rfloor \right) & \textbf{si } p_i \neq 0 \end{cases}$$

$$S_i^{(\boldsymbol{s},\boldsymbol{c},r,\boldsymbol{p})} = \begin{cases} S_i^{(\boldsymbol{s},\boldsymbol{c},r)} & \textbf{si } p_i = N_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{p})} - 1 \\ O_i^{(\boldsymbol{P})} + T_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{p})} \left((p_i + 1) + \left\lfloor \dfrac{I_i^{(\boldsymbol{c},\boldsymbol{s},r)} - O_i^{(\boldsymbol{P})}}{T_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{p})}} \right\rfloor \right) & \\ & \textbf{si } p_i \neq N_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{p})} - 1 \end{cases}$$

4. Cette valeur est valable pour la partie 1 du standard JPEG2000, mais, dans la partie 2, les coordonnées peuvent prendre l'une des deux valeurs 0 ou 1.

5. La taille des codes-blocs de la section 4.4 ne dépend que de la sous-image correspondante.

Et les bornes de ce même pavé dans la sous-bande $\boldsymbol{b}$ valent :

$$\begin{aligned} I_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{b},\boldsymbol{p})} &= \left\lceil \frac{I_i^{(\boldsymbol{c},\mathbf{s})} - 2^{r-1} b_i}{2^r} \right\rceil \\ & \qquad\qquad r \in \{1, 2, \ldots, \mathfrak{D}^{(\boldsymbol{c},\boldsymbol{s})}\} \\ S_i^{(\boldsymbol{c},\boldsymbol{s},r,\boldsymbol{b},\boldsymbol{p})} &= \left\lceil \frac{S_i^{(\boldsymbol{c},\mathbf{s})} - 2^{r-1} b_i}{2^r} \right\rceil \end{aligned}$$

4.1.1.3. *Intérêt*

Le fait d'avoir introduit les concepts de canevas et de position unique pour chaque échantillon – quels que soient sa composante, sa résolution, sa sous-image et son pavé – permet de le manipuler de manière uniforme.

Il est alors possible de créer des partitions, de découper, de transposer ou de pivoter l'image lorsqu'elle est sous sa représentation multirésolution.

Dans la plupart des applications, chaque composante est décrite par une seule sous-image contenant plusieurs partitions en pavés pour répondre au besoin de performance en compression lors de la création du flux binaire (section 4.5).

En fonction des choix faits en termes de résolutions et de composantes, certaines sous-images et certains pavés peuvent être vides sans que cela ne gène en quoi que ce soit. Ils sont tout simplement identifiés au travers de leurs bornes, et, donc, automatiquement rejetés lors des manipulations ultérieures.

Le lecteur intéressé par cette notion de canevas et aux possibilités qu'elle offre, est invité à lire le chapitre 11 du livre de D. S. Taubman et M. W. Marcellin [TAU 02].

4.1.2. *Le centrage des échantillons*

La transformée en ondelettes fournit une collection de sous-bandes séparant les hautes fréquences des basses fréquences, tout en conservant leurs positions spatiales (voir chapitre 2 volume 1).

Les échantillons des hautes fréquences ont, par nature même des ondelettes, une distribution symétrique et centrée en 0. Cependant, les basses fréquences qui proviennent de filtrages passe-bas, ne respectent pas ce critère. Les valeurs de ces échantillons sont donc translatées pour avoir une moyenne nulle suivant la règle :

$$I'(x, y) = I(x, y) - 2^{M-1},\ \forall x, y$$

avec $I(x, y)$ les échantillons – de l'image ou de la sous-image – représentés par des entiers *non signés* de dynamique 2^M.

En complément, l'approche irréversible normalise les échantillons dans l'intervalle unitaire centré à l'origine : $[-1/2, +1/2]$.

4.1.3. *Les espaces multicomposantes*

JPEG2000 traite les composantes d'une image indépendamment les unes des autres. Ainsi, chaque composante possède sa propre dynamique et ses propres dimensions. Cependant, les trois composantes des images couleurs, classiquement les canaux rouge (R), vert (G) et bleu (B), sont fortement corrélées. Afin de séparer au mieux la luminance de la chrominance, le standard JPEG2000 définit une transformée optionnelle sur les trois premières composantes. Pour appliquer la transformée, les trois premières composantes doivent forcément partager la même dynamique et les mêmes dimensions.

Deux transformées couleurs sont alors proposées :

1) une transformée réversible (*Reversible Color Transform RCT*) ;

2) une transformée irréversible (*Irreversible Color Transform ICT*).

4.1.3.1. *La transformation réversible : RCT*

Cette transformation est utilisée principalement pour de très hautes qualités de restitution de l'image après décompression. Il s'agit alors d'opérer une transformation sans perte. L'application directe et sa réciproque sont définies par :

1) la transformation directe s'écrit :

$$\left\{ \begin{array}{lcl} x_Y[\mathbf{n}] & = & \lfloor \frac{x_R[\mathbf{n}]+2x_G[\mathbf{n}]+x_B[\mathbf{n}]}{4} \rfloor \\ x_U[\mathbf{n}] & = & x_B[\mathbf{n}] - x_G[\mathbf{n}] \\ x_V[\mathbf{n}] & = & x_R[\mathbf{n}] - x_G[\mathbf{n}] \end{array} \right.$$

2) la transformation inverse s'écrit :

$$\left\{ \begin{array}{l} x_G[\mathbf{n}] = x_Y[\mathbf{n}] - \lfloor \frac{x_U[\mathbf{n}]+x_V[\mathbf{n}]}{4} \rfloor \\ x_R[\mathbf{n}] = x_V[\mathbf{n}] + x_G[\mathbf{n}] \\ x_B[\mathbf{n}] = x_U[\mathbf{n}] + x_G[\mathbf{n}] \end{array} \right.$$

4.1.3.2. *La transformation irréversible : ICT*

Elle est irréversible, car les coefficients de l'application et de sa réciproque sont à valeurs réelles. On a donc un effet de *quantification* dû à l'approximation faite de ces valeurs. Ce modèle est celui déjà utilisé dans JPEG, mis à part que les composantes de chrominance ne sont plus sous-échantillonnées. L'application directe et sa réciproque sont définies par les trois coefficients :

$$\alpha_R \triangleq 0,299 \quad \alpha_G \triangleq 0,587 \quad \alpha_B \triangleq 0,144$$

et les relations :

$$\begin{aligned} x_Y[\mathbf{n}] &= \alpha_R x_R[\mathbf{n}] + \alpha_G x_G[\mathbf{n}] + \alpha_B x_B[\mathbf{n}] \\ x_{Cb}[\mathbf{n}] &= \frac{0,5}{1-\alpha_B}(x_B[\mathbf{n}] - x_Y[\mathbf{n}]) \\ x_{Cr}[\mathbf{n}] &= \frac{0,5}{1-\alpha_R}(x_R[\mathbf{n}] - x_Y[\mathbf{n}]) \end{aligned}$$

Sa version matricielle, où les coefficients sont des valeurs approximatives de l'application, permet de linéariser le procédé :

1) la transformation directe s'écrit :

$$\begin{pmatrix} x_Y[\mathbf{n}] \\ x_{Cb}[\mathbf{n}] \\ x_{Cr}[\mathbf{n}] \end{pmatrix} = \begin{pmatrix} 0,299000 & 0,587000 & 0,114000 \\ -0,168736 & -0,331264 & 0,500000 \\ 0,500000 & -0,418688 & -0,081312 \end{pmatrix} \begin{pmatrix} x_R[\mathbf{n}] \\ x_G[\mathbf{n}] \\ x_B[\mathbf{n}] \end{pmatrix}$$

2) la transformation inverse s'écrit :

$$\begin{pmatrix} x_R[\mathbf{n}] \\ x_G[\mathbf{n}] \\ x_B[\mathbf{n}] \end{pmatrix} = \begin{pmatrix} 1,0 & 0,0 & 1,402000 \\ 1,0 & -0,344136 & -0,714136 \\ 1,0 & 1,772000 & 0,0 \end{pmatrix} \begin{pmatrix} x_Y[\mathbf{n}] \\ x_{Cb}[\mathbf{n}] \\ x_{Cr}[\mathbf{n}] \end{pmatrix}$$

4.2. Les transformées en ondelettes

JPEG2000 repose sur deux outils performants en complexité algorithmique, en temps d'exécution et en qualité d'image reconstruite après décompression. Le premier outil est le sujet de cette section et concerne la représentation multirésolution. Le second est le codage du flux binaire. Il est décrit dans les sections 4.4 et 4.5.

Les définitions et propriétés de la théorie des ondelettes étant expliquées dans le chapitre 2 du volume 1, on fournira simplement l'aspect technique des deux transformations en ondelettes discrètes choisies pour la partie 1 du standard JPEG2000.

Pour les mêmes raisons que lors du prétraitement des images multicomposantes, l'analyse multirésolution requiert deux approches différentes. Une approche doit être réversible pour ne pas engendrer de pertes. L'autre doit être irréversible, afin d'atteindre des taux de compression plus élevés, au prix de pertes significatives.

4.2.1. *La transformée irréversible*

Les filtres autorisés par la partie 1 du standard sont ceux de CDF 9/7. CDF désigne les auteurs des filtres : Cohen, Debauchies et Feauveau. 9/7 indique la taille des filtres

d'analyse : neuf échantillons pour le filtre passe-bas $h = [\tilde{h}_{-4}, \tilde{h}_{-3}, \tilde{h}_{-2}, \tilde{h}_{-1}, \tilde{h}_0, \tilde{h}_1, \tilde{h}_2, \tilde{h}_3, \tilde{h}_4]$ et sept pour le filtre passe-haut $g = [\tilde{g}_{-3}, \tilde{g}_{-2}, \tilde{g}_{-1}, \tilde{g}_0, \tilde{g}_1, \tilde{g}_2, \tilde{g}_3]$. Les valeurs approximatives de ces filtres sont [ACH 04] :

$$\begin{cases} & & \tilde{h}_0 & = & +0.602949018236358 \\ \tilde{h}_{-1} & = & \tilde{h}_1 & = & +0,266864118442872 \\ \tilde{h}_{-2} & = & \tilde{h}_2 & = & -0,078223266528988 \\ \tilde{h}_{-3} & = & \tilde{h}_3 & = & -0,016864118442875 \\ \tilde{h}_{-4} & = & \tilde{h}_4 & = & +0,026748757410810 \end{cases}$$

$$\begin{cases} & & \tilde{g}_0 & = & +1.115087052456994 \\ \tilde{g}_{-1} & = & \tilde{g}_1 & = & -0,591271763114247 \\ \tilde{g}_{-2} & = & \tilde{g}_2 & = & -0,057543526228500 \\ \tilde{g}_{-3} & = & \tilde{g}_3 & = & +0,0912717631142495 \end{cases}$$

Les filtres de synthèse s'écrivent en fonction des filtres d'analyse : $\overline{h}_n = (-1)^n \tilde{g}_n$ et $\overline{g}_n = (-1)^n \tilde{h}_n$. Pour plus d'efficacité, l'implémentation procède suivant le principe du *schéma lifting* [ACH 04, ACH 06].

4.2.2. *La transformée réversible*

Dans sa version réversible, les filtres de Le Gall 5/3 ont été choisis pour la partie *part-1* du standard. Les valeurs des filtres d'analyse sont [ACH 04] :

$$\begin{cases} & & \tilde{h}_0 & = & +\frac{3}{4} \\ \tilde{h}_{-1} & = & \tilde{h}_1 & = & +\frac{1}{4} \\ \tilde{h}_{-2} & = & \tilde{h}_2 & = & -\frac{1}{8} \end{cases} \text{ et } \begin{cases} & & \tilde{g}_0 & = & +1 \\ \tilde{g}_{-1} & = & \tilde{g}_1 & = & -\frac{1}{2} \end{cases}$$

Ces filtres peuvent également servir pour une compression avec pertes, bien qu'il ait été vérifié expérimentalement que le banc CDF 9/7 offrait une meilleure qualité visuelle à des taux de compression identiques.

Pour une compression sans perte, l'intérêt des filtres de Le Gall est de fournir des coefficients entiers, donc, de ne pas requérir à la quantification qui est sujette à des pertes. Comme pour la transformée irréversible, l'implémentation adopte le *schéma lifting*.

4.3. La quantification

En mode réversible aucune quantification n'est évidemment appliquée. En revanche, le mode irréversible fournit des coefficients réels qui sont à quantifier. Mais,

quel que soit le mode utilisé – réversible ou pas – le codeur et le décodeur doivent s'accorder sur le nombre maximum de bits nécessaires au codage des valeurs, et sur le nombre maximal de bits à utiliser pour parer à tout débordement de l'intervalle initial des valeurs des coefficients ($[-2^{-1}, 2^{-1}]$).

4.3.1. *Le mode irréversible*

Pour la quantification des valeurs réelles des coefficients d'ondelettes, l'algorithme dit *à zone morte* est utilisé (voir chapitre 3 volume 1). Son principe est celui du quantificateur scalaire habituel en mettant l'accent sur les valeurs hautes comme l'illustre la figure 4.4. Il fournit les valeurs quantifiées :

$$q_b[\mathbf{n}] = \text{sign}(x_b[\mathbf{n}]) \left\lfloor \frac{|x_b[\mathbf{n}]|}{\Delta_b} \right\rfloor$$

où :

– $\Delta_b \propto 2^{-\epsilon_b}$ est le pas de quantification associé à la sous-bande b ;

– $x_b[\mathbf{n}]$ est un échantillon de la sous-bande b de valeur comprise entre -2^{-1} et 2^{-1} ;

– sign$(x_b[\mathbf{n}])$ prend la valeur -1 ou $+1$ suivant que l'échantillon est négatif ou positif.

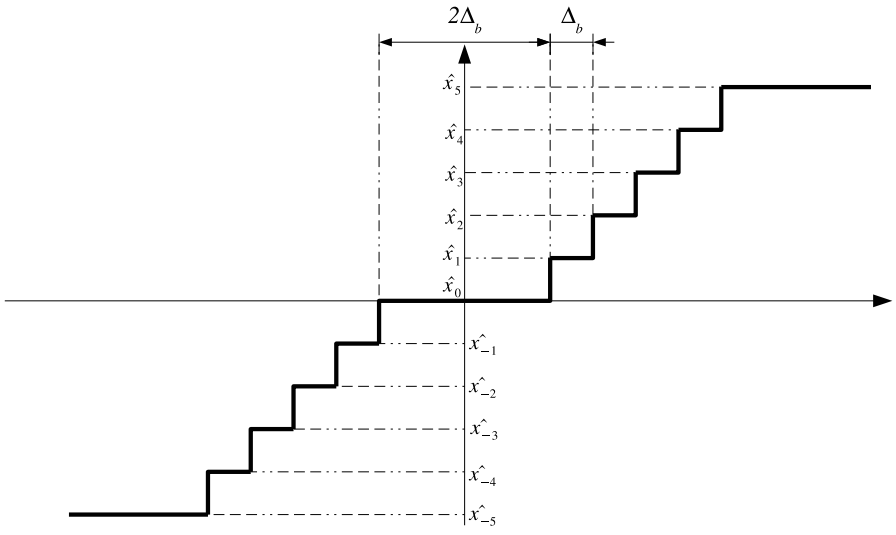


Figure 4.4. *Illustration du quantificateur à zone morte*

Pour parer toute éventualité, le codeur et le décodeur tolèrent un léger dépassement de l'intervalle $[-2^{-1}, 2^{-1}]$ à l'aide d'un paramètre G [TAU 02] :

$$-2^{G-1} < x_b[\boldsymbol{n}] < 2^{G-1}, \forall b \tag{4.1}$$

où $G \in [0, 7]$ avec $G = 1$ une valeur classique.

La valeur quantifiée $q_b[\mathbf{n}]$ est alors décomposée en son signe $\chi_b[\mathbf{n}]$ et son module $v_b[\mathbf{n}]$:

$$\chi_b[\mathbf{n}] = \text{sign}(x_b[\mathbf{n}]) \textbf{ et } v_b[\mathbf{n}] = \left\lfloor \frac{|x_b[\mathbf{n}]|}{\Delta_b} \right\rfloor$$

Ainsi redéfini, le signe $\chi_b[\boldsymbol{n}]$ requiert un seul bit. En revanche, le nombre de bits K_b, nécessaire pour coder le module $v_b[\boldsymbol{n}]$, doit être estimé. Suivant l'équation (4.1), le module $v_b[\boldsymbol{n}]$, est borné supérieurement :

$$v_b[\boldsymbol{n}] = \left\lfloor \frac{|x_b[\boldsymbol{n}]|}{\Delta_b} \right\rfloor < 2^{\varepsilon_b + G - 1}$$

et le nombre maximal de bits à allouer à $v_b[\boldsymbol{n}]$ vaut alors :

$$K_b^{\max} = \max(0, \varepsilon_b + G - 1)$$

Par ailleurs, le pas de quantification doit s'adapter aux différents niveaux de résolutions offerts par l'analyse multirésolution. Pour ce faire, Δ_b est défini à l'aide d'un exposant ϵ_b et d'une mantisse μ_b [TAU 02] :

$$\Delta_b = 2^{-\varepsilon_b} \left(1 + \frac{\mu_b}{2^{11}}\right)$$

avec $0 \leq \varepsilon_b < 2^5$ et $0 \leq \mu_b < 2^{11}$.

L'exposant 11 provient du fait que les valeurs des paramètres ε_b et μ_b sont rangées sur 11 bits dans les marqueurs du flux binaire. La représentation hiérarchique dyadique, induite par la transformée en ondelettes, permet une écriture récurrente de l'exposant et du module [TAU 02] :

$$\varepsilon_b = \varepsilon_0 + (d_b - d_0) \text{ et } \mu_b = \mu_0$$

où :

– ε_0 et μ_0 sont les valeurs associées à la sous-bande d'approximation de plus basse résolution ;

– d_b et d_0 sont les compteurs du nombre de décomposition à effectuer pour obtenir les sous-bandes d'indices b et $\mathfrak{D}(\boldsymbol{c}, \boldsymbol{s})$ respectivement.

Ainsi seules les valeurs ε_0 et μ_0 doivent être enregistrées; les autres valeurs sont calculées à partir de celles-ci.

Connaissant Δ_b, la déquantification est alors immédiate :

$$\hat{x}[\boldsymbol{n}] = \begin{cases} 0 & \textbf{si } v_b[\boldsymbol{n}] = 0 \\ \chi_b[\boldsymbol{n}](v_b[\boldsymbol{n}] + \frac{1}{2})\Delta_b & \textbf{sinon} \end{cases}$$

REMARQUE 4.4.– *Comme le codeur peut n'envoyer qu'une partie du code $\hat{v}_b[\boldsymbol{n}]$, (voir section 4.4), le décodeur peut ne connaître que la valeur v_b tronquée des $p_i[\boldsymbol{n}]$ bits de poids faible non réceptionnés :*

$$\hat{x}_b[\boldsymbol{n}] = \begin{cases} 0 & \textbf{\textit{si }} v_b[\boldsymbol{n}] = 0 \\ \hat{\chi}_b[\boldsymbol{n}](\hat{v}_b[\boldsymbol{n}] + \frac{1}{2})\Delta_b & \textbf{\textit{sinon}} \end{cases}$$

4.3.2. *Le mode réversible*

Si le codeur fonctionne en mode réversible, alors la transformée fournit des valeurs entières qui ne sont pas quantifiées :

$$q_b[\mathbf{n}] = x_b[\mathbf{n}]$$

avec :

$$-2^{B-1+X_b+G} < x_b[\mathbf{n}] < 2^{B-1+X_b+G}$$

où :

– B est le nombre de bits pour un échantillon de la sous-bande b,

– X_b dépend du type de sous-bande [TAU 02] :

$$X_{LL^{(d)}} = 0, X_{LH^{(d)}} = X_{HL^{(d)}} = 1 \text{ et } X_{HH^{(d)}} = 2, \forall d$$

Afin d'homogénéiser les deux modes, l'exposant vaut :

$$\varepsilon_b = B + X_b$$

Ceci permet d'avoir un calcul de K_b^{max} analogue au calcul en mode irréversible :

$$K_b^{\max} = \max(0, \epsilon_b + G - 1)$$

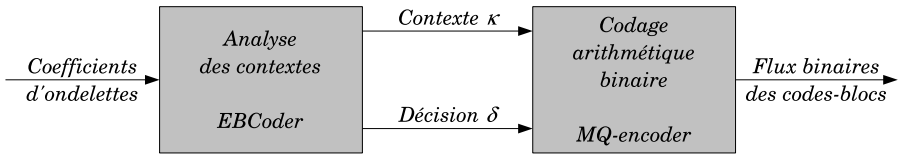


Figure 4.5. *Représentation schématique de EBCOT Tier-1*

4.4. Tier-1 : codage

Comme l'illustre la figure 4.5, il s'agit principalement d'un codage contextuel *EBCoder* (*Embedded Bloc Coder*). EBCoder opère non pas sur les coefficients, mais, sur les plans binaires de ces coefficients et fournit en résultat des couples (κ, δ), de contextes (κ) et de décisions (δ). Les contextes indiquent l'état des bits et de leurs voisinages directs.

Ensuite, le codeur *MQ-encoder* transcrit les séquences de couples (κ, δ) en flux binaires compressés.

EBCoder est dit *à blocs embarqués* (*embedded bloc coder*), car l'information codée pour un échantillon fait référence, à la fois, à la valeur de l'échantillon et à celles des échantillons du bloc $\mathcal{B}$ auquel l'échantillon appartient. Le schéma adopté entre dans la catégorie des algorithmes incrémentiels, ce qui a pour effet de minimiser la mémoire nécessaire et de permettre une forte parallélisation lors d'une implémentation matérielle [ACH 04].

Son résultat est abstrait, car il ne fournit pas un flux binaire de données compressées, mais, un flux de symboles qui renseigne sur les bits à codifier et leurs voisinages (voir figure 4.3b). Reste alors à coder, effectivement, cette séquence de symboles en un flux binaire compressé. C'est le rôle de l'algorithme de codage arithmétique binaire *MQ-encoder* (voir section 4.4.5).

Le rapport entre déformation et débit a été le sujet d'une étude théorique et expérimentale poussée. Les lignes générales sont présentées ici. Le lecteur intéressé par ce sujet se référera au chapitre 12 du livre de David Taubman et Michael W. Marcellin [TAU 02] et au chapitre 7 du livre de Tinku Acharya et Ping-Sing Tsai [ACH 04].

L'idée générale revient à constater que plus la déformation entre l'image originale et l'image reconstruite diminue, plus le nombre de bits utiles à la reconstruction augmente; et donc le débit. Il faut donc toujours jongler entre la distorsion et le débit.

Lorsque le débit varie, les valeurs optimales du rapport déformation-débit, forment l'enveloppe convexe contenant l'ensemble des valeurs sous-optimales (voir figure

4.6a). David S. Taubman démontre [TAU 00, TAU 02] plusieurs propriétés liées à cette enveloppe.

PROPOSITION 4.1.– *Si, pour les valeurs à coder, les plans binaires sont traités indépendamment les uns des autres, alors les mesures de déformation-débit, faites à la fin du codage de chaque plan binaire, sont des valeurs optimales.*

PROPOSITION 4.2.– *Pour le codage de chaque plan de bits, il existe des points de césure qui permettent de mieux approcher l'enveloppe convexe.*

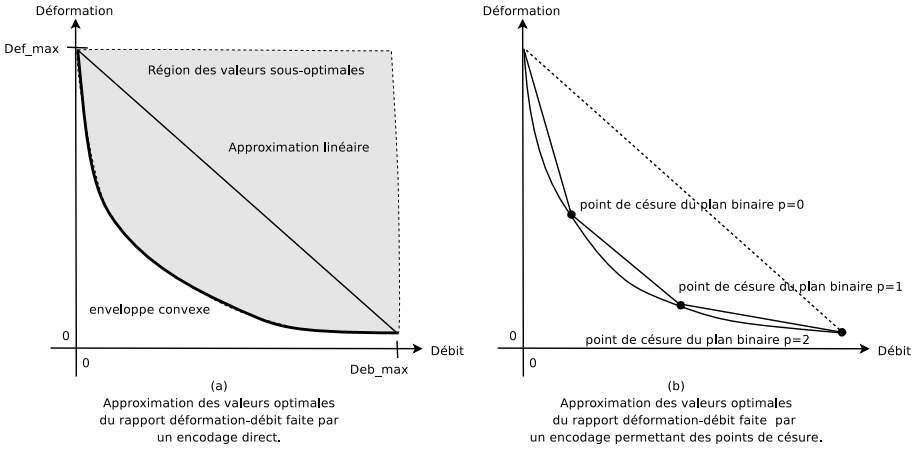


Figure 4.6. *Approximation de l'enveloppe convexe définie par les valeurs optimales du rapport déformation-débit. (a) Cette approximation ne tient compte que des points avant l'envoi et après la réception de tous les bits. (b) Le codage, en plans binaires, introduit des points de césures qui améliorent l'approximation.*

Autrement dit, l'enveloppe convexe peut être mieux approchée par les segments de droites reliant les mesures de déformation-débit des plans binaires p et $(p+1)$ (voir figure 4.6b). De plus, lorsque ces segments sont brisés, – en appliquant les *procédures de fractionnement* (voir paragraphe 4.4.2) – l'approximation en est encore améliorée (voir figure 4.7). Les points délimitant les segments sont appelés les *points de césure*.

Toujours dans un souci d'optimisation, les coefficients de la transformée en ondelettes ne sont pas codés en séquences, ligne par ligne, mais, organisés au sein de chaque sous-bande.

4.4.1. *Organisation des échantillons*

Les échantillons de la description multirésolution sont ordonnés suivant leurs niveaux de résolution. En premier, se présentent les échantillons du niveau D de

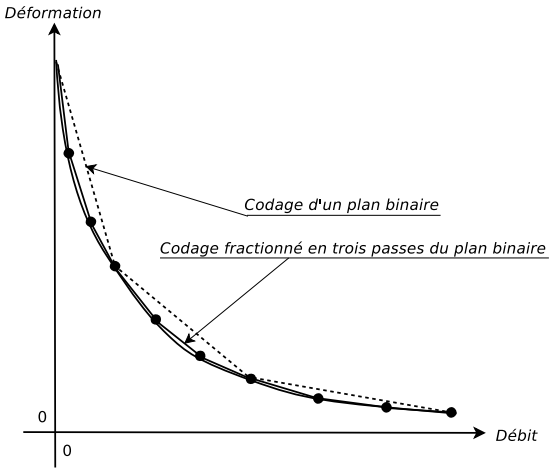


Figure 4.7. *En fractionnant en trois passes successives le codage en plan binaire, l'approximation est encore améliorée*

résolution, c'est-à-dire l'image d'approximation basse résolution LL_D, puis, ceux du niveau supérieur de résolution qui regroupe les images de détails $HL^{(D)}$, $LH^{(D)}$ et $HH^{(D)}$, suivis des échantillons des images $HL^{(D-1)}$, $LH^{(D-1)}$ et $HH^{(D-1)}$, du niveau de résolution suivant et ainsi de suite. La figure 4.8 illustre le regroupement en niveaux de résolution des images de détails.

Ensuite, chaque image, ou sous-bande, est découpée en *codes-blocs* de taille (32×32) ou (64×64). Ces tailles ont été choisies, après expérimentation, comme présentant les meilleures caractéristiques en termes de rapport déformation-débit.

Pour augmenter l'efficacité du codage, chaque code-bloc est découpé en *bandelettes, d'une largeur égale à celle des* blocs et d'une hauteur de quatre lignes comme l'illustre la figure 4.9. Ces bandelettes permettent d'améliorer les performances lorsque plusieurs échantillons successifs non significatifs sont observés. Ces cas particuliers se retrouvent lors du codage des répétitions par la procédure RLC ; le parcours au sein des bandelettes se faisant en colonnes (voir figure 4.10), les répétitions détectées ne concernent que les configurations de 4 bits consécutifs, sur une même colonne.

Tout au long du codage, la notion de voisinage direct[6] d'un échantillon est utilisée. Un voisinage peut naturellement déborder sur une bandelette voisine, quand l'échantillon concerné est en bordure de bandelette (voir figure 4.10).

6. Les huit plus proches voisins.

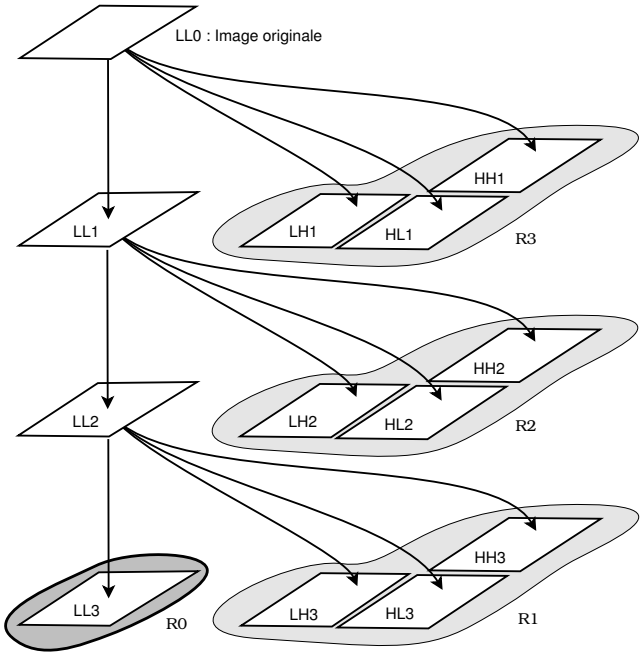


Figure 4.8. *Regroupement des sous-bandes en niveaux de résolution (D=3)*

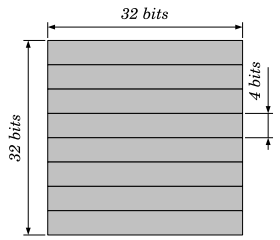


Figure 4.9. *Découpage d'un code-bloc* 32×32 *en bandelettes*

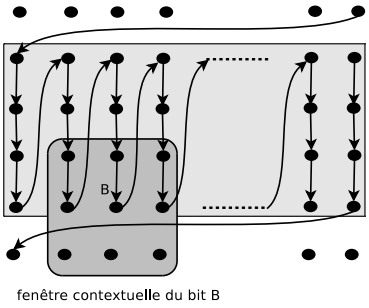


Figure 4.10. *Structure d'une bandelette et du voisinage contextuel d'un pixel*

Il y a bien une interdépendance des bandelettes au sein d'un même code-bloc. Mais, les codes-blocs restent indépendants les uns des autres. Ainsi, pour un échantillon en bordure de code-bloc, les voisins extérieurs au code-bloc prendront la valeur nulle.

4.4.2. *Codage en plans binaires fractionnés*

Chaque code-bloc est traité indépendamment des autres codes-blocs. Le traitement est séquentiel, suivant les plans binaires des échantillons, en partant du plan binaire de poids fort. Chaque plan binaire p est parcouru suivant l'ordre défini par les bandelettes qui le constituent. L'algorithme exécute, dans l'ordre indiqué, les trois procédures suivantes :

1) *procédure de propagation des valeurs significatives* (*Significant Propagation Pass* : SPP). cette étape code les bits dont les valeurs ne sont pas encore significatives, mais, dont au moins un voisin direct l'est. Autrement dit, l'algorithme suppose que cette valeur va devenir significative. Si son bit au plan p vaut 1, alors le signe est également codé et les voisins deviennent à leur tour des candidats à cette étape, à moins qu'ils n'aient déjà été détectés comme significatifs. D'où le terme de propagation des valeurs significatives ;

2) *procédure d'affinage des modules* (*Magnitude Refinement Pass* : MRP). il s'agit de coder les bits pour les échantillons détectés significatifs lors du traitement d'un des plans précédents q (avec $q > p$). Les valeurs significatives sont alors améliorées par ce traitement ;

3) *procédure de récupération* (*CleanUp Pass* : CUP). Cette dernière étape va récupérer l'ensemble des valeurs qui n'ont pas été traitées par l'une des deux étapes précédentes. Les échantillons détectés sont alors forcément non significatifs. A nouveau, les voisins directs sont utilisés. Pour un échantillon donné, si son bit au plan p vaut 1, alors son signe est également codé. Cette étape va également repérer les séquences de quatre bits nuls en colonne au sein d'une bandelette pour les traiter à l'aide de la procédure RLC.

REMARQUE 4.5.– *Seule la troisième procédure, CUP, est effectuée pour le premier plan binaire présenté –le plan binaire de poids fort – car toutes les valeurs sont non significatives initialement.*

4.4.2.1. *Pourquoi un codage fractionné ?*

Dans un codage direct, où chaque point de césure correspond au codage d'un plan binaire, l'approximation de l'enveloppe convexe du rapport déformation-débit s'écarte assez fortement du contour optimal comme le montre la figure 4.6b. En revanche, en fractionnant la codification suivant les trois étapes SPP, MRP et CUP, l'approximation est nettement améliorée, comme l'illustre la figure 4.7.

4.4.2.2. *En quoi le codage est embarqué ?*

Comme décrit par la suite, le codage d'un bit d'un échantillon ne code pas seulement sa valeur, mais, également la topologie des valeurs significatives dans son voisinage. En effet, les bits d'un échantillon ne sont pas codés en une unique séquence. Mais, au contraire, ils sont répartis dans le flux généré pour n'apparaître qu'au moment opportun. Autrement dit, les bits de poids fort ont la priorité au codage sur les autres bits. Le contexte d'un bit permet de savoir à quel moment le coder.

Ces trois procédures – SPP, MRP et CUP – sont décrites en détail dans les annexes A.2, A.3 et A.4. L'annexe A.1 définit les variables d'état utilisées. Ces variables étant également présentes dans l'algorithme général, elles sont décrites dans la section suivante.

4.4.3. *Les variables d'état*

Il est important de ne pas oublier que le traitement s'effectue sur un code-bloc $\mathcal{B}$ indépendamment des autres codes-blocs.

Un échantillon $q[\mathbf{n}]$ est décrit par son signe $\chi[\mathbf{n}]$et par son module $v[\mathbf{n}]$. La variable $v^p[\mathbf{n}]$ représente le bit du plan p du module $v[\mathbf{n}]$. p^{max} désigne le plan du bit de poids fort de ce module. De plus, les procédures SPP, MRP et CUP, partagent les variables d'état :

1) une variable de *détection* $\sigma[\mathbf{n}]$ est initialisée à 0 et prend la valeur 1 dès que le premier bit non nul de $v[\mathbf{n}]$ est codé. (Quand l'indice $\mathbf{n}$ est hors du code-bloc, sa valeur est considérée nulle) ;

2) une variable d'*affinage* $\gamma[\mathbf{n}]$ passe à 1 quand l'opérateur de raffinement du module (*magnitude refinement coding* MRC) vient d'être appliqué pour le bit du plan p. Sinon elle vaut 0 ;

3) une variable de *sélection* $\eta[\mathbf{n}]$ est initialisée à 0 à chaque nouveau plan p prêt à être codé. Il passe à 1 lorsque la valeur $v^p[\mathbf{n}]$ est susceptible de devenir significative. Autrement dit, $\eta[\mathbf{n}]$ passe à 1 dans la procédure SPP quand $v^p[\mathbf{n}]$ n'est pas encore significatif, mais, qu'au moins un de ses voisins l'est.

Chaque codage est représenté par une séquence de couples (κ, δ) qui indique le contexte (κ) et la décision (δ) associés au bit de position $\mathbf{n}$ dans le plan binaire p. Le contexte codifie l'état des voisins du bit observé.

4.4.4. *L'algorithme général*

Comme on l'a constaté précédemment, le plan binaire de poids fort n'est pas candidat aux deux étapes SPP et MRP. De manière analogue, le plan binaire de poids

faible ne présente pas d'échantillon susceptible d'intéresser l'étape CUP. Les traitements de ces deux plans particuliers sont donc extraits de la boucle principale qui opère les trois étapes SPP, MRP et CUP dans cet ordre. Cette procédure est effectuée pour l'ensemble des bandelettes constituant le code-bloc en cours de traitement.

EMBEDDED_BLOC_CODING($\mathcal{B}$)
1 initialiser σ et γ à 0 pour toute la bandelette
2 initialiser η à 0 pour le plan $p^{\max}$
3 positionner $\boldsymbol{n}$ au début du plan $p^{\max}$
4 CLEAN_UP_PASS($\mathcal{B}$)
5 **pour** $p = p^{\max} - 1$ **à** 2
6 **faire** initialiser η à 0 pour tout le plan p
7 positionner $\boldsymbol{n}$ au début du plan p
8 SIGNIFICANT_PROPAGATION_PASS($\mathcal{B}$)
9 positionner $\boldsymbol{n}$ au début du plan p
10 MAGNITUDE_RAFINMENT_PASS($\mathcal{B}$)
11 positionner $\boldsymbol{n}$ au début du plan p
12 CLEAN_UP_PASS($\mathcal{B}$)
13 initialiser η à 0 pour tout le plan de poids faible
14 positionner $\boldsymbol{n}$ au début du plan 1
15 SIGNIFICANT_PROPAGATION_PASS($\mathcal{B}$)
16 positionner $\boldsymbol{n}$ au début du plan 1
17 MAGNITUDE_REFINMENT_PASS($\mathcal{B}$)

L'architecture développée à travers ces algorithmes est de type *pipe-line* parallèle. Chaque code-bloc est indépendant des autres. Il peut donc être traité sur un processeur dédié. Au sein du traitement d'un code-bloc, le codage des couples (contexte, décision) est itératif. Ainsi, la transmission est réellement embarquée. Reste à détailler la procédure MQ_ENCODE utilisée pour générer le flux binaire.

4.4.5. *Codage arithmétique binaire*

Le codeur EBCODER fournit les flux de couples *(contexte, décision)* que le codeur arithmétique binaire (MQ_ENCODER) va transformer en flux binaires. Le schéma de fonctionnement du codeur MQ est une version proche de celle du codeur QM déjà utilisé par le standard JBIG2 (*Joint Bilevel Image processing Group*) pour le codage d'images binaires (voir chapitre 3 et annexe C du volume 1). Ce dernier a été introduit pour répondre au besoin de compression temps réel que requiert le fac-similé. Déjà, la notion de contexte est utilisée.

L'idée générale du codeur MQ est de transformer les données binaires – ici les valeurs de décision – en deux catégories :

– les symboles les plus probables (*Most Propable Symbols* : MPS) ;

– les symboles les moins probables (*Least probable Symbols* : LPS).

Cette idée peut s'expliquer en prenant pour exemple une image binaire – à valeurs 0 pour les pixels noirs et 1 pour les pixels blancs. Dans une région noire, le bit 0 est le plus probable. Il sera donc transformé en un MPS. En revanche, le bit 1 sera transformé en un LPS.

Inversement, dans une région blanche, le bit 1 est le plus probable. Il sera transformé en un MPS et le bit 0 sera transformé en un LPS. Pendant le décodage, le bit qui vient d'être décodé est alors un MPS ou un LPS et, *en fonction du contexte*, retrouvera sa valeur initiale.

Le fonctionnement global du codeur MQ est illustré en figure 4.11.

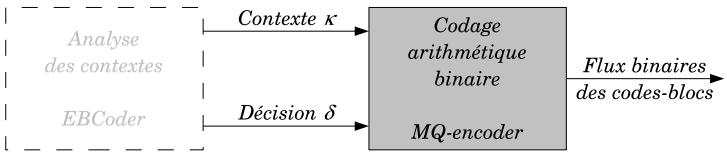


Figure 4.11. *Le codeur MQ*

REMARQUE 4.6.– *Les codeurs arithmmétiques binaires sont décrits en annexe B du volume 1.*

L'algorithme du codeur MQ se présente sous la forme des deux procédures : MQ_CODING_INIT() et MQ_ENCODE(κ, δ). La procédure MQ_CODING_INIT() met à zéro le registre B, affecte la valeur 0,75 à T et initialise un compteur C_T à 12 ou à 13 suivant que le codage précédent a généré une retenue ou pas. Ce compteur C_T indique le nombre maximum de bits pouvant être décalés lors des normalisations des variables T et B. Le tableau 4.1 est également chargé afin d'initialiser l'index, $I(\kappa)$, et les MPSs des contextes κ.

Contexte κ	Index $I(\kappa)$	MPS
0	4	0
1-16	0	0
17	3	0
18	46	0

Tableau 4.1. *Les* 19 *contextes du codeur EBCod et leurs index*

L'algorithme de la procédure MQ_ENCODE(κ, δ) est le suivant :

```
MQ_ENCODE(κ, δ)
1  si δ = MPS(κ)
2     alors CODE_MPS(κ)
3     sinon CODE_LPS(κ)
4  FLUSH()
```

Si la décision δ correspond à la probabilité $\mathrm{MPS}(\kappa)$, alors la procédure CODE_MPS est appelée. Dans le cas contraire, c'est la procédure CODE_LPS qui l'est. La procédure FLUSH transfert le dernier octet dans le flux de sortie. Si cet octet est plein, il faut mémoriser $(C_T = 12)$ qu'il reste un bit de retenue à coder lors du prochain appel de la procédure MQ_ENCODE(κ, δ). Les mises à jour de la variable Q, des probabilités NIMPS et NILPS et de l'indicateur de permutation SWITCH sont faites suivant les valeurs données par le tableau 4.2

4.5. Tier-2

Grâce à JPEG2000, l'utilisateur peut contrôler à la fois le niveau de résolution et le niveau de qualité voulus. Le niveau de résolution est une conséquence directe de l'utilisation de la transformée en ondelettes – réversible ou irréversible. En revanche, le niveau de qualité est lié à la méthode utilisée pour arranger les flux binaires des étapes SPP, MRF et CUP[7]. Pour faciliter les explications qui vont suivre, on nomme ces flux binaires des *codes-flux* en référence aux codes-blocs dont ils découlent.

Définir la meilleure organisation possible des codes-flux et gérer le supplément d'informations qui en découle, est l'objectif de la deuxième phase, *Tier-2*, maintenant décrite.

La variable $L_i^{(j)}$ est le nombre d'octets enregistrés depuis le début dans le code-flux de l'ensemble des codes-blocs $\mathcal{B}_\rangle$ de la sous-bande $\boldsymbol{b}$ jusqu'à la fin de la $j^{\text{ème}}$ passe.

Une approche simple et globale de contrôle de la qualité consiste à regrouper, en premier, les codes-flux ($L_i^{(1)}$ octets) de la première passe pour l'ensemble des codes-blocs $\mathcal{B}_i$ de la sous-bande $\mathbf{b}$, puis à regrouper ceux de la deuxième passe ($L_i^{(2)} - L_i^{(1)}$ octets) et ainsi de suite. La figure 4.12 présente le paquet entrelacé final.

7. Les flux contextuels des étapes SPP, MRP et CUP sont codés en flux binaires à l'aide du MQ-codeur.

I(κ)	Q	NIMPS	NILPS	Switch	I(κ)	Q	NIMPS	NILPS	Switch
0	0,503937	1	1	1	23	0,199245	24	21	0
1	0,304715	2	6	0	24	0,164088	25	22	0
2	0,140650	3	9	0	25	0,140650	26	23	0
3	0,063012	4	12	0	26	0,128931	27	24	0
4	0,030053	5	29	0	27	0,117212	28	25	0
5	0,012474	38	33	0	28	0,105493	29	26	0
6	0,503937	7	6	1	29	0,099634	30	27	0
7	0,492218	8	14	0	30	0,063012	31	28	0
8	0,421904	9	14	0	31	0,057153	32	29	0
9	0,328153	10	14	0	32	0,050561	33	30	0
10	0,281277	11	17	0	33	0,030053	34	31	0
11	0,210964	12	18	0	34	0,024926	35	32	0
12	0,164088	13	20	0	35	0,015404	36	33	0
13	0,128931	29	21	0	36	0,012474	37	34	0
14	0,503937	15	14	1	37	0,007347	38	35	0
15	0,492218	16	14	0	38	0,006249	39	36	0
16	0,474640	17	15	0	39	0,003044	40	37	0
17	0,421904	18	16	0	40	0,001671	41	38	0
18	0,328153	19	17	0	41	0,000847	42	39	0
19	0,304715	20	18	0	42	0,000841	43	40	0
20	0,281277	21	19	0	43	0,000206	44	41	0
21	0,234401	22	19	0	44	0,000114	45	42	0
22	0,210964	23	20	0	45	0,000023	45	43	0
					46	0,503937	46	46	0

Tableau 4.2. *Tableau des mises à jour*

Le terme de flux embarqué prend tout son sens ici, car, plus le point de césure choisi par le décodeur sera tardif, plus la qualité au sein de chaque code-bloc augmente. On contrôle alors le critère de qualité à l'aide de cette organisation des codes-flux. C'est le schéma organisationnel utilisé par la plupart des codeurs embarqués (en 2008). Il présente l'intérêt d'être déterministe, donc efficace. Cependant, le critère de débit n'est à aucun moment pris en compte dans ce schéma. A l'opposé, on peut organiser le paquet en entrelaçant les code-flux suivant le critère de déformation-débit optimal.

Parmi l'ensemble, $\{h_i^0, h_i^1, \cdots\}$, des points de césure du code-bloc $\mathcal{B}_i$ – fournit le codeur EBCoder[8] – David Taubman [TAU 02] montre que la sélection des points optimaux[9] peut se faire en suivant les points de césure h_i^k minimum au sens de la pente

8. Ce sont ceux délivrés par les trois passes SPP, MRP et CUP.

9. Dans une approche plus rigoureuse, il s'agit de points sous-optimaux puisque nous avons un ensemble fini dénombrable de points de césure.

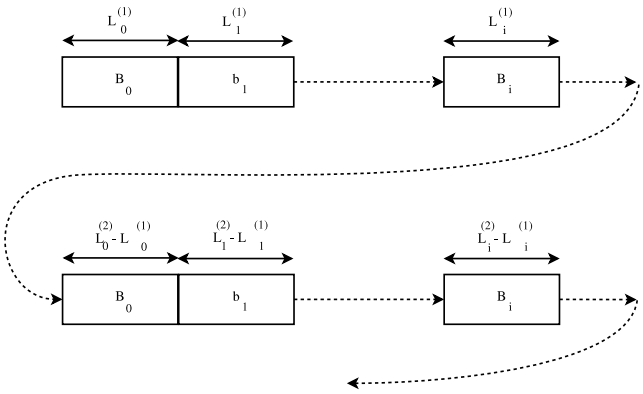


Figure 4.12. *Organisation globale des codes-flux*

λ_i du rapport *déformation/longueur*[10] :

$$\lambda_i(h_i^k) = \min_{h_i^u < h_i^k} \frac{D_i^{(h_i^u)} - D_i^{(h_i^k)}}{L_i^{(h_i^k)} - L_i^{(h_i^u)}}$$

où $D_i^{(h_i^u)}$ est la déformation de la reconstruction due à la coupure du flux au point de césure h_i^u; $L_i^{(h_i^u)}$ est la longueur du flux tronqué au point de césure h_i^u. Ainsi $\left(D_i^{(h_i^u)} - D_i^{(h_i^k)}\right)$ caractérise la diminution relative de la déformation, et $\left(L_i^{(h_i^k)} - L_i^{(h_i^u)}\right)$ informe sur l'augmentation relative de la longueur codée. Si h_i^k est le point optimal pour le code-bloc $\mathcal{B}_i$, alors les points de césure optimaux pour les autres codes-blocs $\mathcal{B}_j$ doivent respecter :

$$\lambda_i(h_i^{k+1}) \geq \lambda_j(h_j^k), \forall j \neq i$$

Mais, la mise en œuvre d'une telle organisation demande d'accompagner chaque code-flux d'informations complémentaires permettant d'identifier le code-bloc associé. Par conséquent, plus l'image est de taille imposante, plus les performances en termes de débit se dégradent.

Entre ces deux extrêmes, David Taubman [TAU 02] propose un compromis en termes de couches de qualité (*quality layers*). Les codes-flux sont partitionnés suivant leurs valeurs de pente $\lambda_i(h_i^k)$: au sein d'une classe de la partition, les valeurs de

10. Les points de césure sont temporellement ordonnés. Et donc écrire $(h_i^u < h_i^k)$, indique l'ensemble des points de césure h_i^u qui sont apparus avant h_i^k.

pente sont similaires. Ces classes forment les couches de qualité proposées par David Taubman. Les codes-flux appartenant à une même couche sont simplement organisés suivant leurs indices de codes-blocs. Ainsi, aucune information complémentaire n'est nécessaire pour identifier de quels codes-blocs proviennent les codes-flux. Le standard JPEG2000 reprend ce principe en proposant trois scénarios :

1) monocouche (*Single Layer*). Si la déformation n'est pas un critère important, alors les codes-flux sont rangés en une unique liste en commençant par les codes-flux du premier code-bloc $\mathcal{B}_1$, puis ceux du deuxième, etc. Ceci est illustré en figure 4.13 ;

2) multicouche spécifique (*Targeted Layers*). Certaines applications spécifient un ensemble restreint de débits autorisés. Cet ensemble sert alors à construire les couches ;

3) Multicouche générique (*Generic Layers*). Si aucune information de la sorte n'est *a priori* connue, mais, que la qualité de la reconstruction est un critère important, le paquet sera divisé en plusieurs couches. Les limites intercouches proviennent d'un ensemble de seuils de pente déformation-longueur qui, judicieusement choisi, permet de partitionner les codes-flux. Expérimentalement, pour des débits variant de 0,05 à 2,0 bits par échantillon, il a été montré qu'un ensemble de cinquante couches avec des codes-blocs de taille 64×64 répondait au critère [TAU 02]. La figure 4.14 en donne une illustration.

Figure 4.13. *Organisation monocouche*

Les couches de qualité utilisées par JPEG2000 forment ainsi un niveau supplémentaire d'abstraction permettant de séparer le rôle du codage – qui lui-même se décompose en une partie codage contextuel et une partie codage entropique – du rôle de l'ordonnancement des données compressées.

La phase *Tier-2* codifie également les informations supplémentaires liées à la représentation multicouche :

– l'information d'appartenance (*inclusion information*) qui indique, pour chaque couche, quels codes-blocs y contribuent ;

– la longueur des codes-flux correspondants ;

– les plans binaires de poids fort auxquels aucun échantillon d'un code-bloc donné ne participe (les échantillons de ce code-bloc sont de modules plus faibles que ceux d'autres codes-blocs) ;

– les points de césure délimitant les couches qui informent sur le nombre de passes SSP, MRP et CUP utilisées au sein de chaque couche ;

– si une couche donnée ne contient aucun code-bloc.

Le dernier cas est codé sur un simple bit. Il est mis à 0, si aucun code-bloc n'est inclus dans la couche concernée. Ce bit est rangé dans l'en-tête associé à la couche. L'information d'appartenance opère suivant deux modes.

Si le code-bloc observé a déjà été introduit dans une couche précédente, un simple bit mis à 1 indique que ce code-bloc est également inclus dans la couche courante.

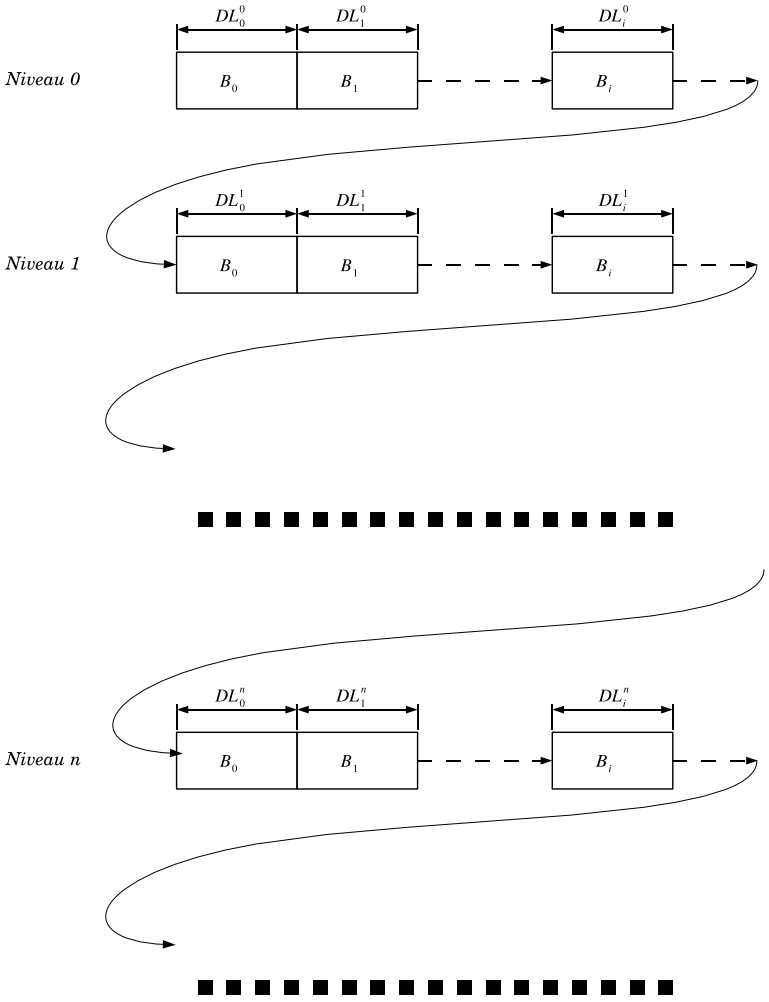


Figure 4.14. *Organisation multicouche générique*

Dans le cas où le code-bloc est inclus pour la première fois, une structure particulière, appelée *Tag-Tree*, codifie l'information pour cette couche et les suivantes. La structure *Tag-Tree* sera étudiée dans la prochaine section. Les plans binaires non significatifs sont également codés à l'aide des *Tag-Tree* dans l'en-tête du pavé (*precinct*) du code-bloc. Le nombre de passes est codifié suivant une variante du codage de Huffman. Enfin, la longueur des codes-flux est codée par deux valeurs; la première indique le nombre de bits utilisés pour coder la longueur et la deuxième indique la valeur proprement dite.

4.5.1. *Tag-Tree*

Un *Tag-Tree* est une version modifiée du *Quad-Tree* permettant de coder, sans redondance, des matrices de valeurs positives. La figure 4.15 montre un exemple de *Tag-Tree*, fourni par la recommandation ISO/IEC JTC1/SC29 WG1 N 1646R et par Tinku Acharya et Ping-Sing Tsai dans [ACH 04].

Le niveau 3 représente les valeurs initiales. Pour le niveau 2, les minimums par groupe de quatre valeurs du niveau précédent (niveau 3) sont codés :

$$\begin{aligned}
q_2(0,0) &= \min\{q_3(0,0), q_3(1,0), q_3(0,1), q_3(1,1)\} = 1 \\
q_2(1,0) &= \min\{q_3(2,0), q_3(3,0), q_3(2,1), q_3(3,1)\} = 1 \\
q_2(2,0) &= \min\{q_3(4,0), q_3(5,0), q_3(4,1), q_3(5,1)\} = 2 \\
q_2(0,1) &= \min\{q_3(0,2), q_3(1,2)\} = 2 \\
q_2(1,1) &= \min\{q_3(2,2), q_3(3,2)\} = 2 \\
q_2(2,1) &= \min\{q_3(4,2), q_3(5,2)\} = 1
\end{aligned}$$

Le procédé est répété pour construire les niveaux 1 et 0 :

$$\begin{aligned}
q_1(0,0) &= \min\{q_2(0,0), q_2(1,0), q_2(0,1), q_2(1,1)\} = 1 \\
q_1(1,0) &= \min\{q_2(2,0), q_2(2,1)\} = 1 \\
q_0(0,0) &= \min\{q_1(0,0), q_1(1,0)\} = 1
\end{aligned}$$

La procédure de codification basée sur ce *Tag-Tree* est assez simple. Chaque nœud est codé par une séquence de bits à 0 terminée par un bit à 1. La longueur de la séquence, hormis le bit à 1, indique la différence des valeurs entre le nœud courant et son parent. Pour la racine, la valeur 0 est prise comme valeur parent. Suivant ce schéma, le nœud $q_3(0,0)$ a pour code 01111 : la différence entre la racine et sa valeur parent (0) est donc de 1. Sa séquence est alors 01. Ensuite, les valeurs étant égales entre enfants et parents, les séquences ne sont composées d'aucun bit à 0. Lors du codage de $q_3(1,0)$, il n'est plus utile de coder les différences entre les niveaux $(-1,0)$, $(0,1)$ et $(1,2)$ qui sont déjà enregistrées avec $q_3(0,0)$. Il reste donc à coder la différence entre les niveaux 2 et 3 : $q_3(1,0)$ prend pour code 001. Le procédé est itéré pour les valeurs suivantes. Ainsi $q_3(2,0)$ est codé 101, $q_3(0,2)$ est codé 011, etc. Le processus est développé en figure 4.16.

4.6. Synthèse

Le standard JPEG2000 est construit sur plusieurs éléments-clés. La transformation en ondelettes, un codeur en deux passes dont la première identifie les contextes plan de bit par plan de bit.

La transformée en ondelettes offre, intrinsèquement, une description multirésolution de l'image. Ainsi, tout en effectuant une analyse dans l'espace spatio-fréquentiel, plusieurs niveaux de résolution de l'image sont obtenus. L'analyse spatio-fréquentielle délivre des coefficients qui peuvent être plus ou moins atténués suivant l'importance qu'ils ont dans la description de l'image.

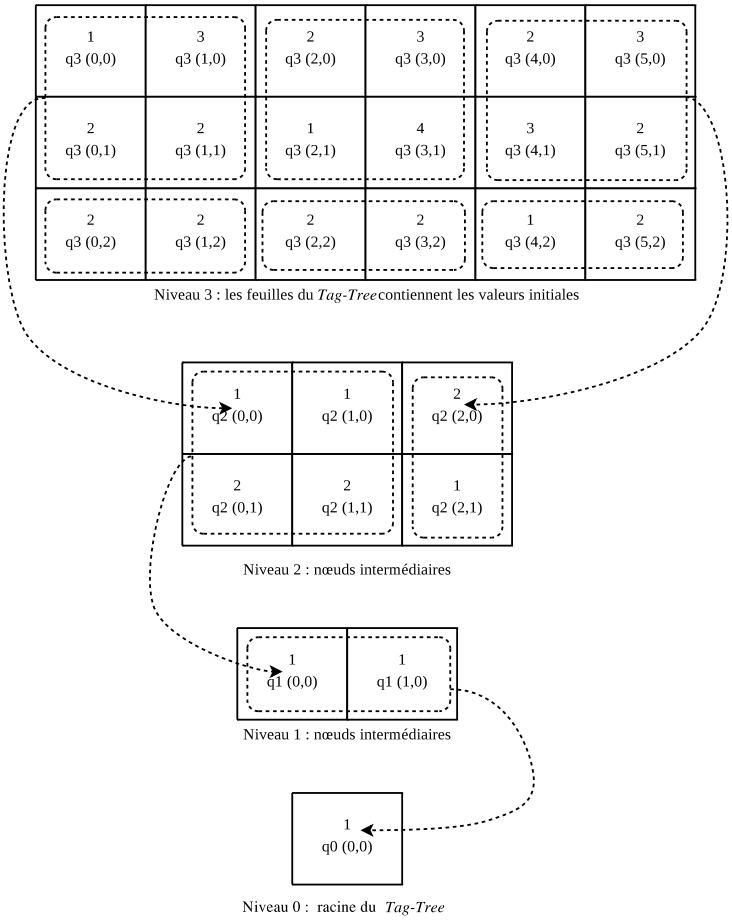


Figure 4.15. *Exemple de* Tag-Tree *d'après ISO/IEC JTC1/SC29 WG1 N 1646R*

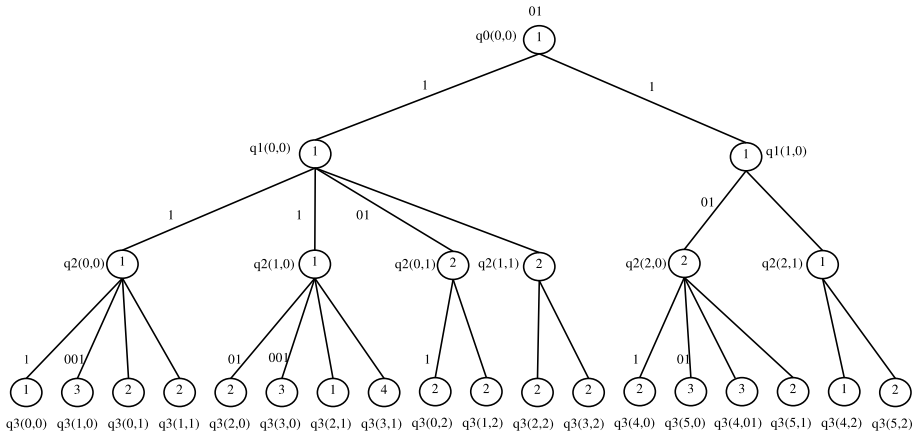


Figure 4.16. *Processus de codage du* Tag-Tree *de la figure 4.15*

On retrouve le même principe qu'avec les coefficients de l'analyse fréquentielle faite par le standard JPEG. En revanche, il n'est plus besoin de découper l'image en imagettes, puisque les coefficients sont localisés dans l'image. Le codeur effectue deux passes afin d'offrir, à tout moment, la meilleure approximation. La première passe, effectue le codage des plans binaires. Puis, la deuxième passe ordonne l'information suivant des critères fournis par l'utilisateur ou le matériel.

Avec ce chapitre se termine la partie consacrée aux techniques et standards de compression d'images. Le premier chapitre présente les compressions sans perte en allant de GIF à JPEG-LS en passant par PNG et *LossLess*-JPEG. Puis, le deuxième chapitre présente le standard JPEG. Celui-ci permet de coder des images, dites naturelles, avec des pertes qui sont, en général, peu visibles. Les taux de compression obtenus permettent une diffusion des images sur Internet. Pour une diffusion sur des réseaux à plus faibles débits et sensibles aux perturbations, le standard JPEG2000 est plus adapté. Il présente des taux de compression nettement supérieurs à ceux de JPEG pour une qualité de rendu de l'image identique. De plus, il permet un débit variable avec un affichage multirésolution. La prochaine partie présente les techniques de compression audiovisuelles où les outils de compression d'images sont réutilisés :

1) DCT sur des bloc de taille 8×8 ;

2) codage DPCM des coefficients DC ;

3) parcours en zigzag des coefficients AC ;

4) transformée en ondelettes discrète ;

5) codage en plans binaires ;

6) etc.

DEUXIÈME PARTIE

Compression et représentation de vidéos

Synopsis

Cette deuxième partie présente les standards MPEG dédiés à la compression audio-visuelle. On y retrouve l'ensemble des outils utilisé pour la compression d'images :

– l'analyse fréquentielle ;

– l'analyse tempo-fréquentielle ;

– la quantification ;

– les codeurs entropiques (Huffman, arithmétique) ;

– etc.

Mais, on y trouve également de nouveaux outils, spécifiques à la compression des audiovisuels :

1) pour la compression d'une vidéo, le temps est un nouvel axe qui sera fortement exploité. Les mouvements des *objets* d'une vidéo sont estimés et codés. Le gain en compression est non négligeable ;

2) pour la compression audio, les signaux sont séparés en deux catégories suivant leurs sources :

a) les *sons synthétiques* sont codés suivant un langage de programmation adéquat : par exemple, le format MIDI,

b) les *sons naturels* sont ceux fournis par un capteur, comme un microphone par exemple. Les outils, présentés au chapitre 4 du volume 1, sont implémentés. En particulier, le signal est observé dans le domaine temps-fréquence et le modèle psychoacoustique est estimé.

C'est volontairement que les normes MPEG sont décrites, dans un ordre historique. Une version de la norme MPEG englobe les versions précédentes. Ainsi, en choisissant une présentation chronologique, les outils sont décrits au fur et à mesure de leur apparition. Cette façon de procéder permet de classer les nombreux outils suivant les applications et les services visés :

1) MPEG1 (chapitre 5) fournit la base de la compression vidéo et audio pour un enregistrement sur CD-ROM ;

2) MPEG2 (chapitre 6) étend MPEG1 à la télédiffusion et aux DVD. Par l'utilisation de réseaux à faibles débits et volatiles (le mobile, par exemple), la télédiffusion requiert des taux de compression plus élevés, ainsi qu'une gestion des erreurs. De nouveaux outils voient alors le jour dont certains sont plus fortement développés dans les versions suivantes : la granularité, les profils et les niveaux ;

3) MPEG4 (chapitre 7) introduit l'interaction et la création multimédia. Les audiovisuels sont structurés en objets, tout comme les langages de programmation sont devenus des *langages orientés objets*. C'est l'évolution majeur de cette version. Elle permet de gagner encore en termes de compression, mais, surtout, d'introduire l'interactivité avec l'utilisateur final à tous les instants de la chaîne : de la production à l'affichage en passant par la diffusion. Les outils de granularité, de profils et de niveaux sont étendus à tous les types d'objets. La séparation des éléments de l'audiovisuel en objets permet de mixer des objets naturels (sons naturels, photographies, vidéos, reconstructions 3D) avec des objets synthétiques (sons et images 2D/3D de synthèse).

L'espace couleur utilisé par MPEG est le YCrCb qui repose sur le principe de la sépération de la luminosité (la composante Y) et de la chrominance (les composantes Cr et Cb). Un convertisseur est embarqué dans tout codeur (resp. décodeur) pour permettre le passage de l'espace RGB à l'espace YCrCb (resp. YCrCb à RGB). Toutefois, le codeur enregistre dans le flux binaire la matrice de passage inverse que les décodeurs doivent utiliser pour la transformation de YCrCb à RGB. Le sous-échantillonnage de la chrominance, utilisé par les normes MPEG, est justifié par des études psychoneuronales de notre vue (voir chapitre 4 du volume 1).

Toutes les normes MPEG définissent uniquement le décodeur et le flux binaire. Peu de contraintes sont définies sur la construction du codeur. Ceci permet au constructeur de développer des codeurs aussi performants qu'ils le souhaitent. Tant que les codeurs fournissent le flux binaire requis, tous les décodeurs peuvent les décoder !

Chapitre 5

MPEG1

Le groupe *Motion Picture Expert Group* (MPEG) a été créé en 1988 avec pour objectif de prendre pleinement possession des capacités offertes par la numérisation des signaux pour la construction de nouveaux supports multimédias regroupant du texte, de l'audio et de la vidéo.

EXEMPLE 5.1.– *Un exemple de codage audiovisuel va permettre de comprendre le besoin de compression. Soit un audiovisuel dont :*

1) la partie vidéo est de résolution 352×288 *avec 24 bits par pixel et avec un taux de rafraichissement*[1] *de 25 Hz ;*

2) la partie audio est stéréo (sur deux canaux) codée sur 16 bits avec une fréquence d'échantillonnage de 44,1 kHz.

Le débit binaire de la partie vidéo est alors de $352 * 288 * 24 * 25 = 60825600\,\text{bits} / \text{sec} \approx 58\,\text{Mb} / \text{sec}$ *et celui de la partie audio de* $2 * 16 * 44100 = 1411200\,\text{bits} / \text{sec} \approx$ *1,35 Mb/sec. La totalité du débit binaire est donc d'environ 60 Mb/sec. Ainsi, pour un film de cent cinq minutes, il faut trouver un support d'environ 370Gb. Ce qui n'est pas envisageable.*

Le codage au format MPEG1 permet l'enregistrement sur un support CD de clips audiovisuels de plusieurs minutes (de quelques centaines de Mo). Il faut donc compresser l'audiovisuel source (ici, avec un taux de quarante).

Pour encore mieux concrétiser l'effort de compression fait, il faut savoir que le débit binaire de l'audiovisuel, au format MPEG1, est identique au débit binaire de la partie audio seule, dans sa version non compressée.

1. Fréquence de défilement des images lors de la présentation.

Le standard MPEG1 s'organise en plusieurs parties dont les trois principales sont :

1) la partie *système*, décrite en section 5.1, (ISO/IEC 11172-1: 1993) ;

2) la partie *vidéo*, décrite en section 5.2, (ISO/IEC 11172-2: 1993) ;

3) la partie *audio*, décrite en section 5.3. (ISO/IEC 11172-3: 1993).

Les standards sont accessibles depuis le site officiel : http://www.chiariglione.org/mpeg/. Le schéma général est donné en figure 5.1. L'audiovisuel, fourni en entrée, est séparé en une source audio et une source vidéo. A partir de ces sources, les parties vidéo et audio du standard se chargent de générer les *flux*[2] audio et vidéo; c'est-à-dire les codes compressés de l'audio et de la vidéo respectivement. La partie système organise le mixage, appelé *multiplexage*, des flux audio et vidéo. Les informations de synchronisation, entre ces deux parties, sont transférées à la partie système.

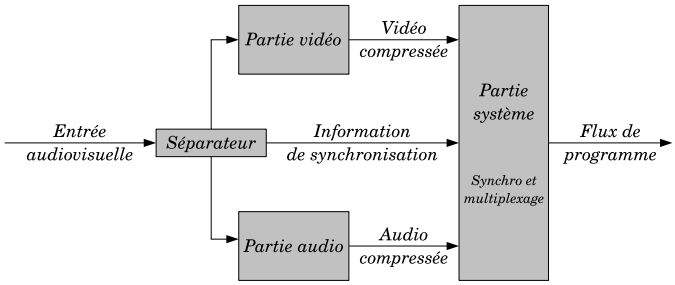


Figure 5.1. *Les trois principales parties du standard MPEG1*

5.1. La partie système

La partie système gère principalement la synchronisation entre le flux audio et le flux vidéo grâce à une horloge. Celle-ci insère, régulièrement, des repères temporels dans le code compressé des deux flux. Ces marques découpent les flux audio et vidéo en sous-séquences que le système organise en paquets, appelés *flux élémentaires empaquetés* (PES[3]). Les marques temporelles sont insérées dans l'en-tête des paquets. Il existe deux types de marques :

1) une DTS (*Decoding Time Stamp*) : une *estampille temporelle de décodage* qui indique quand a été effectué le découpage ;

2) une PTS (*Presentation Time Stamp*) : une *estampille temporelle de présentation* qui indique à quel moment le paquet doit être présenté.

2. *Stream*.

3. *Packetized Elementary Streams*.

Cette organisation en paquets permet également l'accès direct[4] à une sous-séquence de l'audiovisuel.

L'ensemble des paquets est ensuite multiplexé de manière à respecter les contraintes du décodeur. Ce dernier dispose d'une zone mémoire tampon qui lui permet de stocker les paquets à décoder.

Il faut éviter le *débordement* (*overflow*) ; c'est-à-dire que le tampon soit rempli alors qu'un paquet doit y être enregistré. Il faut aussi éviter l'*assèchement* (*underflow*) ; c'est-à-dire que le tampon soit vide alors que le décodeur cherche de nouveaux paquets. Le fonctionnement du décodeur et la taille de son tampon sont spécifiés par le standard. La partie système du codeur organise le multiplexage des paquets en fonction des contraintes fixées par le standard.

Au final, le codeur délivre un *flux de programme* (PS[5]). Ce flux de programme est, habituellement, enregistré sur un support tel que le CD-Rom, le CD-I ou encore le CD-V. La transmission au graveur de CD se faisant généralement *via* un bus PCI sur un ordinateur ou un enregistreur de salon, il est supposé qu'il n'y a pas de perte ou d'erreur lors de la gravure. Aussi, le flux de programme ne contient pas d'information de détection ni de correction d'erreurs.

Un flux de programme correspond à une seule présentation : un film, par exemple.

5.2. La partie vidéo

En entrée, la norme MPEG1 lit une source vidéo au format CIF[6] composée de 352 lignes par image et de 240 (resp. 288) pixels par ligne pour un taux de rafraichissement de 30 Hz (resp. 25 Hz). La différence de fréquences d'échantillonnage est liée au pays de conception et/ou d'utilisation du codeur et du décodeur. Par exemple, en Europe, la fréquence est de 25 Hz, alors qu'elle est de 30 Hz aux Etats-Unis d'Amérique. En sortie, le débit binaire du flux de programme est constant autour de 1,5 Mbits/sec.

Le standard MPEG 1 compresse les sources vidéos en se servant de leurs caractéristiques. Toute vidéo numérique est une séquence temporelle d'images où deux images successives ont une forte probabilité d'avoir beaucoup d'informations en commun. Typiquement, dans un plan séquence (une même prise de vue), deux images proches dans le temps vont partager une grande partie de l'arrière-plan de la scène filmée.

---

4. Aussi appelé *accès aléatoire*.

5. *Program Stream.*

6. *Common Intermediate Format*.

Cette corrélation temporelle est prise en compte par le standard en définissant trois types d'images :

1) une *I-image*, ou image *intra*, est compressée sans tenir compte des images qui la suivent ou qui la précèdent. La compression est donc spatiale, au sein de l'image ;

2) une *P-image*, ou image *prédite*, est prédite à partir d'une image dite de *référence*. L'erreur de prédiction entre l'image courante et l'image de référence est codée. Cette image de référence précède l'image courante dans le temps. Elle est, soit une image *intra*, soit une image *prédite* ;

3) une *B-image*, ou image *bidirectionnelle*, est prédite à partir de deux images de référence. L'erreur de prédiction entre l'image courante et la moyenne des deux images de référence est codée. Ces deux images de référence sont *intra* ou *prédites*. Une référence précède l'image bidirectionnelle tandis que l'autre la succède. L'intérêt d'émettre une double prédiction, à la fois passée et future, est de gérer les occultations et les disparitions des objets ainsi que l'arrivée de nouveaux éléments. La figure 5.2 illustre l'intérêt du codage en B-images.

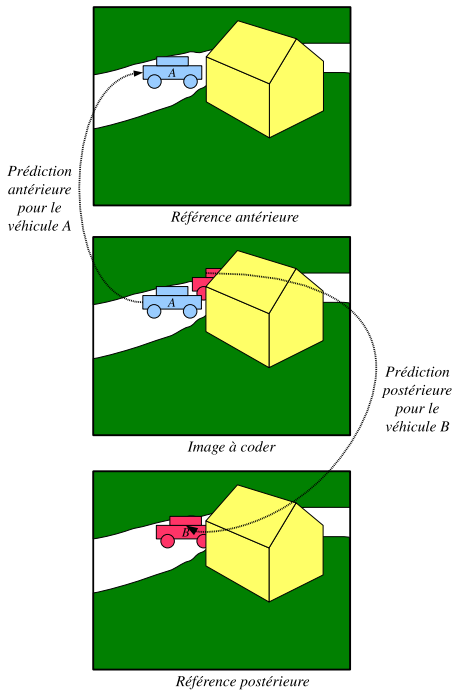


Figure 5.2. *Codage en B-image : le véhicule A, qui se déplace de gauche à droite, est prédit à partir de l'image de référence antérieure, tandis que le véhicule B, qui se déplace de droite à gauche, est prédit à partir de l'image de référence postérieure*

Ces images sont assemblées en *groupes* (GOP[7]). Un groupe d'images est *fermé* quand ses P-images et ses B-images sont décodables en utilisant uniquement les I-images et les P-images qu'il contient. Sinon, le GOP est dit *ouvert*. Deux paramètres N et M, définissent un GOP : N est la taille du GOP; M, la distance entre deux images prédites ou entre une image *intra* et la prochaine image prédite au sein du GOP. Habituellement, N est choisi entre 12 et 15, M entre 1 et 3.

Ces trois types d'images et ces groupes sont illustrés par les deux exemples de la figure 5.3. Du fait de l'utilisation d'images prédites et bidirectionnelles, l'ordre temporel de lecture des images ne peut plus être respecté.

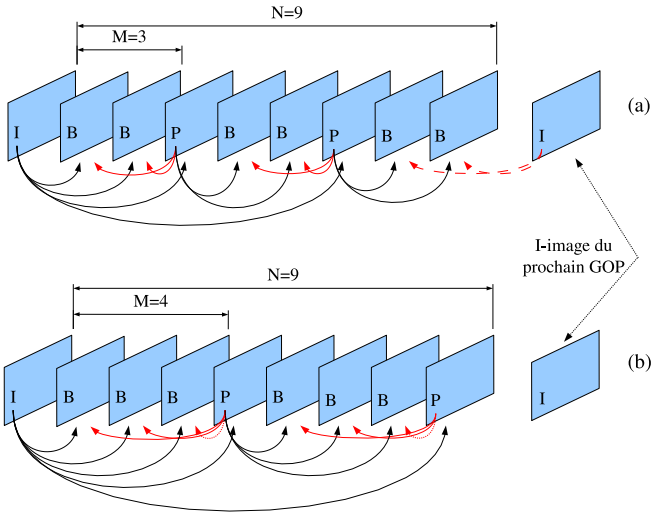


Figure 5.3. *I-image, P-image et B-image : (a) GOP ouvert, (b) GOP fermé*

Dans la figure 5.3a, le décodeur a besoin de reconstruire les images indicées 1 et 4 avant de reconstruire les images indicées 2 et 3. Dans cet exemple, le nouvel ordre des images codées et envoyées au décodeur est :

$$1, 4, 2, 3, 7, 5, 6, 8, 9$$

Puisque le GOP est ouvert, le décodeur doit attendre le prochain GOP pour pouvoir reconstruire les images bidirectionnelles indicées 8 et 9. Ceci est géré par une estampille temporelle de décodage (DTS) fixée par la couche système. Cette estampille DTS est distincte de l'estampille de présentation PTS qui indique à quel moment l'image doit être affichée.

7. *Group Of Pictures*.

Avec l'exemple de la figure 5.3b, le GOP est fermé et l'ordre de codage est :

$$1, 5, 2, 3, 4, 9, 6, 7, 8$$

L'espace couleur du standard MPEG1 est YCrCb avec un sous-échantillonnage *4:2:0 mid-sited* de la chrominance. Ce sous-échantillonnage est illustré en figure 5.4. Les échantillons Cr et Cb de chrominance correspondent à un regroupement en pavés de taille 2×2 de l'échantillonnage initial. Pour chaque pavé, les valeurs de chrominance *Cr* et *Cb* sont estimées à partir des quatre échantillons. De ce fait, la hauteur et la largeur des composantes *Cr* et *Cb* sont la moitié de celles de la composante Y.

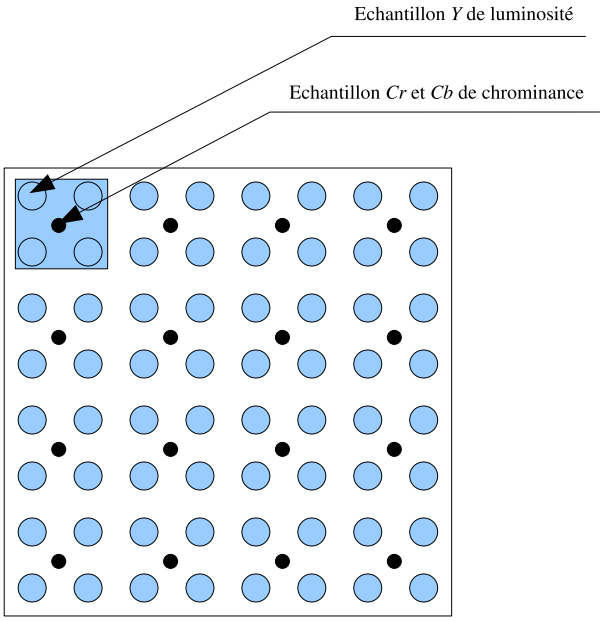


Figure 5.4. *Sous-échantillonnage 4:2:0 mid-sited d'un bloc* 8×8

Ensuite, chaque composante est découpée en bloc de taille 8×8. MPEG1 crée des *macroblocs*, dans le même esprit que JPEG utilise des MCU. Chaque macrobloc regroupe quatre blocs de luminosité (*Y*), un bloc de la composante *Cr* et un bloc de la composante *Cb* (voir figure 5.5). Il n'y a qu'un bloc de la composante *Cr* (resp. *Cb*) pour quatre blocs de la composante *Y*, puisque la chrominance est sous-échantillonnée.

Les macroblocs sont regroupés en *bandes* (*slices*) afin de limiter la propagation des dégradations. Chaque bande est codée indépendamment des autres bandes. Elle possède ses propres paramètres de codage et un marqueur de synchronisation qui indique

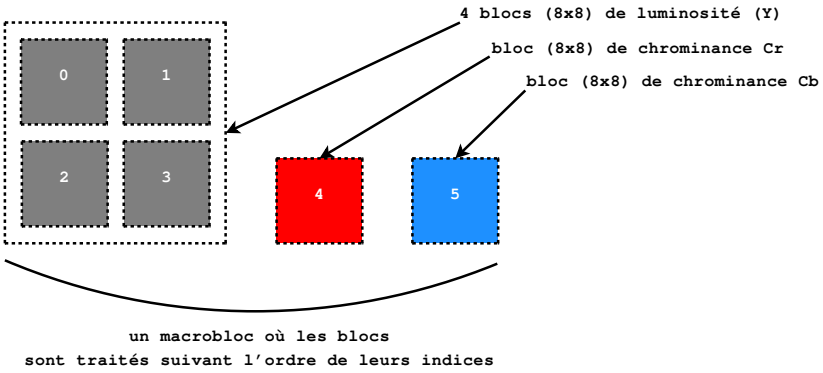


Figure 5.5. *Définition d'un macrobloc*

la position du premier macrobloc. Si une erreur de codage ou une perte de transmission survient sur une bande, le marqueur de synchronisation permet au décodeur de localiser sans erreur les macroblocs de bande suivante. Ainsi, la propagation de l'erreur est-elle limitée. De plus, le débit binaire est contrôlé *via* ces bandes. La figure 5.6 montre un découpage en bandes de macroblocs (seule y est montrée la composante Y avec des macroblocs de taille 16×16).

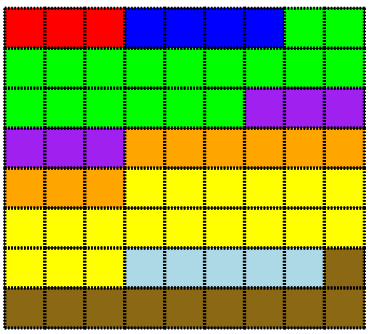


Figure 5.6. *Découpage de l'image en bandes de macroblocs*

5.2.1. *Codage des images* intra *(I-images)*

Pour les images *intra*, le processus de codage est similaire à celui de la norme JPEG (voir section 3.1). Les macroblocs sont traités indépendamment les uns des autres. Au sein d'un macrobloc, les blocs (de taille 8×8) sont analysés séquentiellement suivant l'ordre fixé par les indices de la figure 5.5. L'analyse est faite à l'aide d'une transformée en cosinus discrète (DCT). Le bloc spectral résultant de cette analyse est ensuite quantifié.

Le coefficient DC – de coordonnées $(0, 0)$ – est quantifié suivant un pas, δ, prédéfini :

$$q_{\mathrm{DC}} = \mathrm{round}\left(\frac{\mathrm{DC}}{\delta}\right)$$

où δ dépend de la précision de codage[8] accordée au coefficient DC (voir tableau 5.1).

Nombre de bits	δ
8	8
9	4
10	2
11	1

Tableau 5.1. *Valeurs du pas de quantification δ en fonction de la précision de codage du coefficient DC*

La quantification des autres coefficients (coef. AC) suit la règle suivante :

$$\begin{cases} q_{\mathrm{AC}} & = & \lfloor \frac{\lfloor 16\,\mathrm{AC}\,/S_{\mathrm{AC}} \rfloor + k(\mathrm{AC})M}{2M} \rfloor \\ \widehat{\mathrm{AC}} & = & \frac{2 * q_{\mathrm{AC}} * M * S_{\mathrm{AC}}}{16} \end{cases} \tag{5.1}$$

où :

1) S_{AC} est la valeur issue de la matrice de pondération S, de mêmes coordonnées que le coefficient AC dans le bloc fréquentiel en cours de quantification. La matrice de pondération par défaut est donnée par le tableau 5.2. Elle peut être changé. Dans ce cas, elle doit être inscrite dans le flux binaire pour que le décodeur puisse l'utiliser également ;

2) M est un facteur compris entre 1 et 31. Il est fixé dans l'en-tête de chaque bande. Il peut être changé dans l'en-tête des macroblocs ;

3) la fonction $k(\alpha)$ vaut :

$$k(\alpha) = \begin{cases} 0 & \text{si le macrobloc en cours de quantification provient d'une P-image (voir prochaine section)} \\ \mathrm{sign}(\alpha) & \text{sinon} \end{cases}$$

D'une manière similaire à JPEG, le coefficient quantifié DC est codé en mode prédictif (DPCM). Les coefficients quantifiés AC sont rangés dans une liste construite suivant un parcours en zigzag de la matrice. La liste est codée à l'aide d'un codeur par plages pour les valeurs nulles.

8. C'est-à-dire le nombre de bits alloué par pixel.

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Tableau 5.2. *La matrice de pondération des coefficients S_{AC} par défaut*

Ensuite, l'erreur de prédiction attachée au coefficient DC et la liste des coefficients AC sont transférées au codeur entropique qui est, soit un codeur de Huffman, soit un codeur arithmétique. Pour plus de détails, il faut se référer au chapitre 3 traitant de la norme JPEG.

5.2.2. *Codage des images prédites (P-images)*

Le codage d'une image prédite revient à coder l'erreur engendrée par la prédiction de l'image courante à partir d'une *image de référence*, antérieure dans le temps. Cette image de référence est, soit une I-image, soit une P-image. La prédiction est faite au niveau des macroblocs en deux phases :

1) l'*estimation du mouvement*. Cette première phase cherche à connaître le mouvement *apparent*[9] entre l'image courante et l'image de référence. A chaque macrobloc B, de l'image courante , est associé un macrobloc B_{ref}, de l'image de référence. Le macrobloc de référence est recherché dans une zone précise. Pour un macrobloc de taille $n \times m$, la fenêtre de recherche est de taille $(n + 2p) \times (m + 2p)$. La figure 5.7 montre l'image prédite, le macrobloc à coder, l'image de référence et la zone de recherche. Le centre de la zone de recherche est défini par le centre du macrobloc à coder. La sélection du macrobloc de référence est assurée par la minimisation d'un *critère de dissimilitude*. L'*estimation de mouvement* fournit :

a) le macrobloc B_{ref}, minimisant le critère de dissimilitude,

b) le *vecteur de déplacement* entre la position de B_{ref} et la position (y_B, x_B) du macrobloc B ;

2) la *compensation du mouvement*. La deuxième phase calcule l'*erreur de prédiction* faite par l'estimation du mouvement. Pour chaque macrobloc B, l'*erreur de prédiction* E, avec son macrobloc de référence, B_{ref}, est calculée :

$$E(y, x) = B(y, x) - B_{\mathrm{ref}}(y, x), \forall y, x$$

9. Le mouvement apparent est la projection dans le plan image du mouvement réel des objets composant la scène observée.

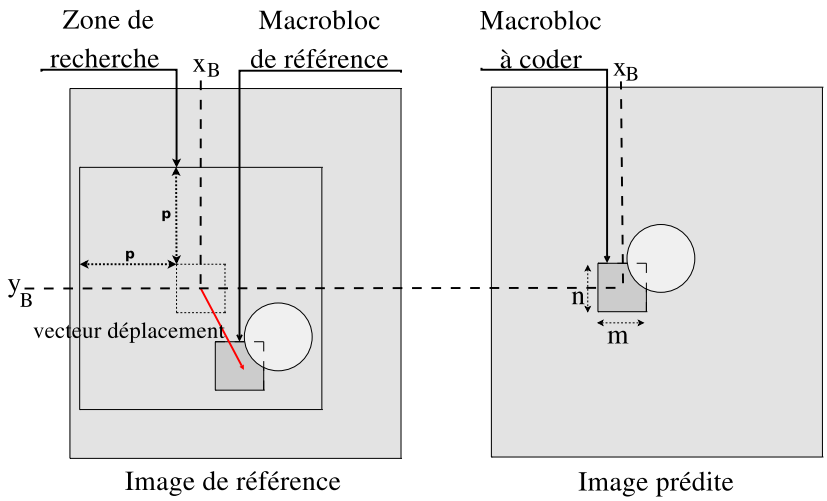


Figure 5.7. *Zone de recherche du macrobloc de référence*

Une fois la compensation de mouvement faite, le macrobloc d'erreur E, est codé suivant le même procédé que celui utilisé pour les macroblocs des I-images. Il est analysé en fréquences à l'aide de la DCT, puis quantifiée (voir équation (5.1)). Le coefficient DC et le vecteur de déplacement sont codés suivant le principe du DPCM. Les coefficients AC sont rangés suivant le parcours en zigzag, puis compressés par un codeur par plage.

Ensuite, un codeur entropique de Huffman ou arithmétique construit le flux binaire. L'ensemble de ces traitements est similaire à celui utilisé par le codeur JPEG (voir chapitre 3). La figure 5.8 illustre l'ensemble du processus de compensation de mouvement.

REMARQUE 5.1.– *L'estimation du mouvement des images couleurs est faite uniquement à l'aide de la composante* Y. *En effet, le mouvement dans les composantes chromatiques ne diffère pas du mouvement observé dans la composante de luminosité. En revanche, les composantes* Cr *et* Cb *interviennent dans la compensation de mouvement.*

REMARQUE 5.2.– *La norme* MPEG1 *utilise, par défaut,* $m = n = 16$ *et* $p = 6$.

REMARQUE 5.3.– *Les annexes B.1 et B.2 fournissent, respectivement, les différents critères de dissimilitude et les algorithmes de recherche du macrobloc de référence, couramment utilisés par les codeurs.*

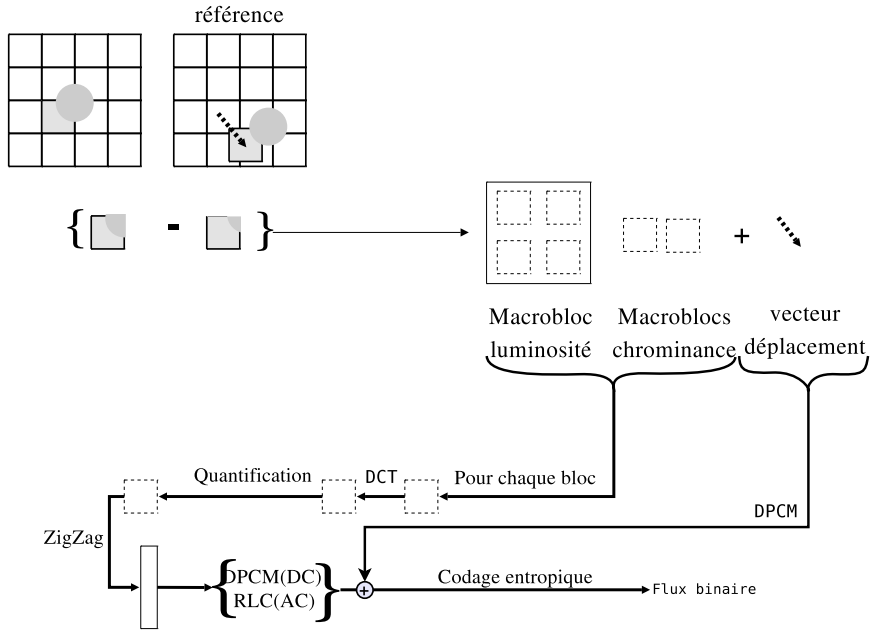


Figure 5.8. *Principe de codage d'un macrobloc prédit*

5.2.3. *Codage des images bidirectionnelles (B-images)*

Le procédé est similaire à celui utilisé pour les images prédites (P-images). Le principe général est décrit par la figure 5.9. La *compensation du mouvement* est faite à l'aide de deux images de référence (P-images ou I-images) au lieu d'une seule pour les images prédites (P-images). L'une des images de référence est temporellement antérieure à l'image bidirectionnelle à coder, tandis que l'autre est postérieure. Ainsi les occultations, disparitions et apparitions dues à des mouvements rapides et/ou à des scènes fortement encombrées sont mieux prises en compte.

L'*estimation du mouvement* est identique à celle utilisée pour les P-images. Cependant, puisqu'il y a deux images de référence, il y a également deux macroblocs de référence $B_{\text{ref}}^{\text{ant}}$ et $B_{\text{ref}}^{\text{post}}$. Ainsi, la prédiction faite pour la compensation de mouvement correspond à la moyenne de ces deux blocs de référence :

$$B_{\text{ref}} = \frac{B_{\text{ref}}^{\text{ant}} + B_{\text{ref}}^{\text{post}}}{2} \tag{5.2}$$

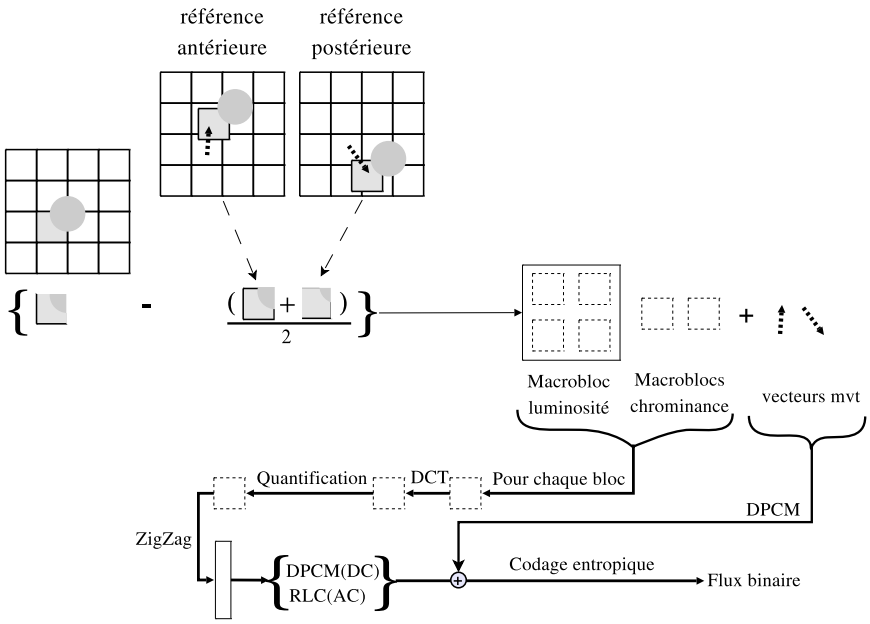


Figure 5.9. *Principe de codage d'un macrobloc bidirectionnel*

Ce résultat n'est pas obligatoirement celui renvoyé par le processus d'estimation de mouvement. Habituellement, pour une B-image, le codeur calcule les valeurs du critère de dissimilitude pour trois configurations possibles :

1) une seule image de référence antérieure ;

2) une seule image de référence postérieure ;

3) le cas général décrit par l'équation (5.2) avec deux images de référence.

Ensuite, le codeur conserve la configuration qui minimise la dissimilitude. Il indique le choix qu'il a fait, dans l'en-tête du flux. Il en résulte qu'un seul vecteur est enregistré pour la configuration 1 ou la configuration 2, alors que deux vecteurs sont enregistrés si la configuration 3 est élue. La prédiction est faite à l'aide d'une seule ou deux images de référence suivant la configuration sélectionnée.

5.3. La partie audio

La compression audio tire profit des études sur le signal temporel et, surtout, sur le système auditif humain. Les techniques utilisées sont donc en partie spécifiques aux signaux audio afin d'effectuer une compression qui n'engendre que des pertes imperceptibles par l'humain (voir chapitre 4 volume 1).

La partie audio est constituée de trois *couches* imbriquées les unes dans les autres[BRA 99, PAN 93, PAN 94, PAN 95]. La couche III, plus connue sous le sigle MP3[10], peut décoder les flux binaires des couches I et II. De même, la couche II reconnaît les flux de la couche I.

Plusieurs modes d'audition sont possibles pour ces trois couches :

– le mode *mono* utilise un seul canal d'enregistrement du signal ;

– le mode *stéréo* sépare le signal en deux canaux, gauche (L) et droit (R) ;

– le mode *dual-mono* est l'enregistrement de deux signaux monophoniques indépendants. Par exemple, l'enregistrement en mono d'un discours sur un canal et sa traduction en une autre langue sur l'autre canal.

De plus la couche III autorise le mode *stéréo mixé* qui fonctionne soit par *mixage en amplitude* soit par *mixage en moyenne* (de sigle *MS-stereo*[11]) :

1) *mixage en amplitude*. Quand les hautes fréquences, des deux canaux, gauche et droit, ont des enveloppes spectrales identiques, le mixage en amplitude fusionne en un seul signal les valeurs des amplitudes de ces hautes fréquences. Au signal monophonique résultant est adjoint le rapport des amplitudes. Ainsi, le décodeur peut séparer les deux canaux et restituer les signaux stéréophoniques initiaux ;

2) *mixage en moyenne*. Quand les signaux stéréophoniques, L et R, sont très similaires (donc fortement corrélés), le mixage en moyenne construit un signal de moyenne, L+R, et un signal de différence, L-R. Le deuxième signal (celui de différence) étant d'entropie plus faible (forte corrélation des signaux originaux), le débit s'en trouve diminué.

La construction de ces codeurs (resp. décodeurs) audios se fonde sur les études de la perception humaine décrites dans le chapitre 4 du volume 1. La description schématique du codeur commun à ces trois couches est donnée en figure 5.10a :

1) en premier lieu, le codeur effectue une transformation du signal du domaine temporel vers le domaine spectral. Pour ce faire, un banc de filtres découpe le signal en 32 sous-bandes fréquentielles de largeurs identiques. Chaque sous-bande fournit une description spectrale du signal pour une plage donnée de fréquences. Mais, la taille des filtres du banc ne coïncide pas avec la taille des *fenêtres fréquentielles* du système auditif humain : une sous-bande de basses fréquences couvre plusieurs fenêtres fréquentielles. De plus, il faut noter que la transformation et son inverse ne sont pas exactes; le signal reconstuit, après analyse, puis synthèse, à l'aide du banc de filtres – sans autre étape intermédiaire – est une (très bonne) approximation du signal original. Il y a donc des pertes dès cette étape ;

10. Pour MPEG1 *Layer* III.

11. Pour *Middle/Side stereo*.

2) parallèlement au découpage du signal en sous-bandes fréquentielles, celui-ci est analysé pour la construction d'un modèle psychoacoustique. Ce modèle fournit les fonctions de masquage utiles pour la compression proprement dite (voir chapitre 4 volume 1). Grâce aux fonctions de masquage obtenues pour chacune des sous-bandes, la quantification et le codage sont optimaux ;

3) pour accroître la qualité du signal encodé, le débit binaire est variable. Le principe est de supposer que le débit binaire est constant. Puis, quand le codage des échantillons requiert moins de bits qu'autorisé, ceux qui ne sont pas utilisés sont stockés. Les *bits en réserve* peuvent alors être utilisés lorsque des échantillons plus gourmands doivent être codés ;

4) le signal est empaqueté dans le flux binaire.

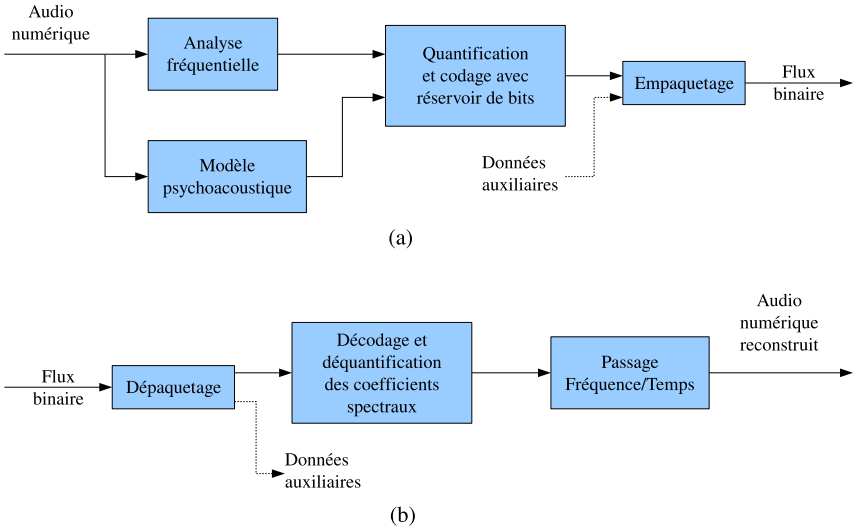


Figure 5.10. *MPEG1 audio : (a) codeur, (b) décodeur*

Comme le montre la figure 5.10b, le décodeur dépaquette le flux binaire, décode les échantillons et déquantifie ces derniers. Une fois les échantillons reconstruits, ils sont envoyés au banc de filtres qui reconstruit le signal audio dans le domaine temporel.

La suite de cette section décrit les principaux modules de la partie audio :

1) l'analyse fréquentielle, qui fait appel à un banc de filtres pour effectuer une transformation temps/fréquences ;

2) le modèle psychoacoustique ;

3) la quantification, le codage et le débit binaire variable ;

4) les spécificités de la couche III.

5.3.1. *L'analyse fréquentielle*

Le signal, dans sa forme temporelle, est découpé en *séquences* grâce aux 32 filtres du banc. Plus la séquence est longue, meilleure est la précision en fréquences. De manière duale (voir chapitre 2 volume 1), plus la séquence est courte, meilleure est la précision en temps.

Ainsi, différentes tailles de séquences sont définies par les trois couches :

– la couche I découpe le signal en séquences de 384 échantillons. A la réception d'une séquence, chaque filtre fournit un *paquet* de douze échantillons fréquentiels : $32 * 12 = 384$;

– les couches II et III opèrent sur des séquences de 1 152 échantillons. Chaque filtre fournit un paquet de 36 échantillons. Un paquet de la couche II ou III correspond à trois paquets de la couche I : $32 * (3 * 12) = 1\,152$.

La figure 5.11 montre le découpage fréquentiel et l'imbriquation des couches I, II et III.

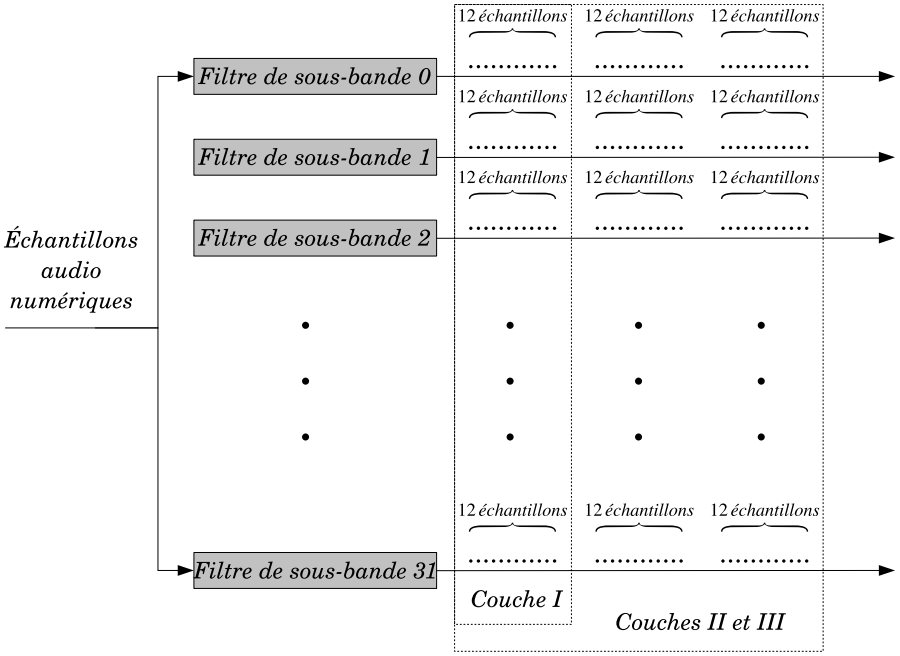


Figure 5.11. *L'analyse fréquentielle par banc de filtres et regroupement des échantillons : la couche I fournit des paquets de douze échantillons alors que les couches II et III fournissent des paquets de 36 échantillons*

5.3.2. *Le modèle psychoacoustique*

En même temps que le signal est découpé en bandes spectrales, le modèle psychoacoustique est construit. Il n'est valide que pour la séquence en cours d'analyse. Aussi, faut-il s'assurer de la synchronisation entre la construction du modèle psychoacoustique et l'analyse fréquentielle du signal. Ce modèle permet d'effectuer, au mieux, la quantification et le codage.

Par exemple, dans le cas de la couche I, le tampon mémoire (*buffer*) en entrée est de taille 512 alors que la séquence analysée par le banc de filtres est de taille 384. Ce *buffer* fonctionnant en mode FIFO[12], il faut attendre que $(512 - 384)/2 = 64$ échantillons soient traités par le banc de filtres pour que le modèle et l'analyse fréquentielle arrivent simultanément à la deuxième étape (quantification et codage).

Le standard ne définit pas de modèle spécifique, mais, fournit deux exemples :

– le *modèle 1*, utilisé par les couches I et II, est le plus simple des deux ;

– le *modèle 2* est utilisé par la couche III.

Dans les deux cas, une transformation fréquentielle est opérée. Celle-ci devant être plus précise que l'analyse fréquentielle délivrée par le banc de filtres, une transformée de Fourier est utilisée.

Au sein de chaque sous-bande, le seuil d'audibilité est ajusté en fonction des amplitudes fortes du signal. Le seuil d'audibilité, par défaut, est estimé expérimentalement (voir chapitre 4 volume 1).

5.3.3. *La quantification, le codage et le débit binaire variable*

Chaque bande fréquentielle est ensuite comparée au seuil qui lui correspond. Ceci fournit une mesure relative de la valeur d'amplitude du signal. Le rapport entre le signal et le seuil d'audibilité permet d'ajuster au mieux le nombre de bits alloués à chaque échantillon. En effet, le seuil étant envoyé dans l'en-tête du flux binaire, seul le rapport de chaque échantillon à ce seuil a besoin d'être codé. Les échantillons en deçà du seuil n'étant pas audibles, ne sont pas enregistrés.

De plus, un *débit binaire constant virtuel* est fixé. Ainsi, lorsqu'il n'est pas atteint, les bits non utilisés sont mémorisés pour être utilisés plus tard. Cette mémorisation des bits permet de dépasser le seuil indiqué par le débit binaire virtuel pour les séquences délicates.

12. *First In First Out*.

5.3.4. *Les spécificités de la couche III*

La couche III utilise le banc de filtres comme le fait la couche II. Ensuite, les échantillons en sortie de chaque sous-bande sont regroupés en blocs courts de six échantillons ou en blocs longs de 18 échantillons. Chaque bloc est alors analysé plus finement en termes de fréquences.

Afin de diminuer les effets de passage d'un bloc au suivant, les blocs sont doublés en taille (12 et 36 échantillons respectivement) avec un recouvrement de 50 % entre deux blocs consécutifs. Par exemple, deux blocs courts consécutifs ont une taille de douze échantillons et partagent six échantillons. La figure 5.12 illustre le recouvrement entre blocs de même taille. Le passage d'un bloc long à un bloc court ou d'un bloc court à un bloc long, se fait progressivement comme le montre la figure 5.13.

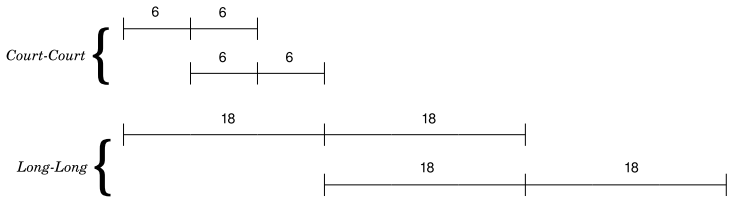


Figure 5.12. *Recouvrement entre blocs longs et blocs courts*

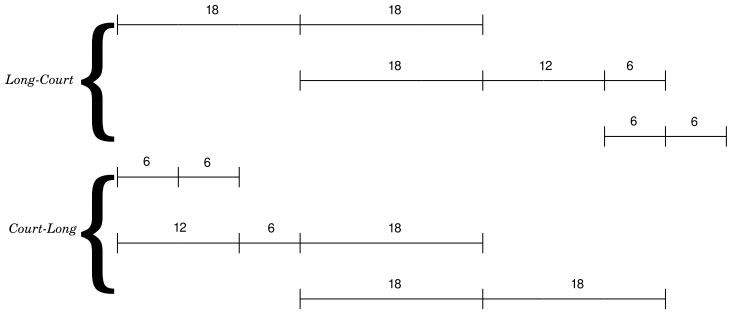


Figure 5.13. *Passage d'un bloc long à un bloc court et réciproquement*

Du fait des recouvrements entre blocs, une version *modifiée* de la DCT (MDCT) est utilisée pour affiner l'analyse fréquentielle au sein de chaque bloc. Elle est suivie d'une suppression des *échos* et *prééchos* sur les blocs longs avant la quantification et le codage. Les prééchos, observables dans le domaine temporel, proviennent des blocs longs qui ont « tendance » à déborder lors de l'analyse.

La MDCT augmente le nombre de sous-bandes de l'analyse fréquentielle en réduisant leurs largeurs. Ainsi, elle apporte une plus grande précision en fréquences au sein de chaque sous-bande. La figure 5.14 fournit le schéma général de la couche III.

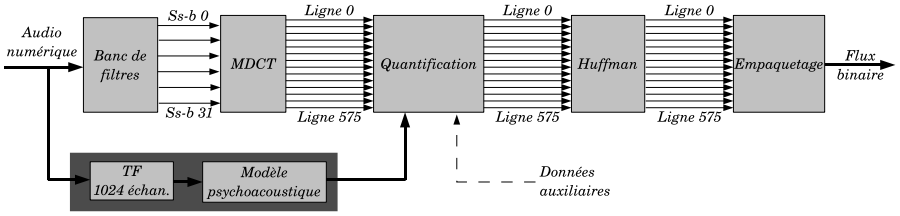


Figure 5.14. *Principe de la couche III*

5.4. Synthèse

Le standard MPEG1 vise à définir un audiovisuel numérique ayant des taux de compression conséquents tout en gardant une qualité vidéo et audio correcte pour un enregistrement sur un cédérom.

Ainsi, MPEG1 a mis en avant l'intérêt de la prédiction et des transformées fréquentielles : DCT, banc de filtres, MDCT, etc. Intérêt confirmé par le succès du format audio MP3.

Bien que les techniques mises en jeu pour la compression de l'audio et de la vidéo soient différentes, le fonctionnement général est similaire :

1) le signal audio (resp. vidéo) est découpé en séquences (resp. en blocs) ;

2) chaque séquence (resp. bloc) est analysée ;

3) la séquence spectrale (resp. bloc spectral), résultant de cette analyse, décrit le signal suivant des caractéristiques proches de la perception humaine. C'est un atout majeur, car l'humain est bien l'utilisateur final des audiovisuels.

Dans sa partie vidéo, MPEG1 utilise les propriétés spatiales et temporelles de la vidéo :

1) la *compression spatiale* est identique à ce que propose déjà le standard JPEG. Les images, qui en résulte, sont appelées des *images intra* ;

2) la *compression temporelle* s'appuie sur les images *intra* pour compresser plus fortement les images restantes. Deux techniques sont proposées :

a) une *image prédite* (P-image) est obtenue en prenant pour image de référence une image *intra* ou prédictive qui la précède dans le temps. Les blocs de l'image à compresser sont prédits à partir des blocs de l'image de référence. Comme pour

tout processus de prédiction, l'erreur de prédiction – supposée faible – est codée. Le déplacement entre le bloc et sa prédiction est également enregistré dans le flux final,

b) une *image bidirectionnelle* est construite suivante le même procédé. Mais, la prédiction s'appuie sur deux images de référence, *intra* ou prédites. Une des images de référence précède temporellement l'image à compresser, tandis que l'autre la suit.

Le format MPEG1 fournit un flux binaire, contenant des images *intra*, prédites et bidirectionnelles, qui convient parfaitement au stockage sur cédérom. Cependant, la compression ne permet pas le stockage et la télédiffusion d'audiovisuels de taille conséquente.

Le format MPEG2, décrit dans le prochain chapitre, propose de remédier à ce défaut. En particulier, il permet la télédiffusion avec des taux de compression efficaces et ajoute le contrôle d'erreurs.

Chapitre 6

MPEG2

En 1991, le standard MPEG1 est apparu pour les disques compacts (CD-Rom). Très vite, ce support est devenu limité. La taille des fichiers multimédias générés pour un film classique dépassait la taille du support. Il a alors fallu s'orienter vers des supports plus adaptés, mais, aussi tenter de réduire la taille de ces fichiers. Une nouvelle norme, MPEG2, a ainsi vu le jour en 1995. Ce standard utilise comme support le disque vidéo numérique (DVD) et la télédiffusion par ondes hertziennes, par câble et par satellite.

MPEG2 reprend les caractéristiques de MPEG1, mais, étend le champ d'investigation. Il propose un format multimédia permettant à l'utilisateur de choisir différentes options tels que la langue de l'audio et les sous-titres. Les prémisses de l'*interaction* entre l'utilisateur et les données multimédias prennent naissance avec ce standard. En particulier, sa partie, intitulée DSM-CC, *booste* notre boîtier TV (*set-top box*, voir glossaire) en le dopant avec de nouvelles fonctionnalités.

Il est composé de plusieurs parties dont les principales sont :

– la partie *système*, décrite en section 6.1, (ISO/IEC 13818-1:2000) ;

– la partie *vidéo*, décrite en section 6.2, (ISO/IEC 13818-2:2000) ;

– la partie *audio*, décrite en section 6.3, (ISO/IEC 13818-3:2000) ;

– la partie *contrôle d'accès* (DSM-CC), décrite en section 6.4, (ISO/IEC 13818-6:2000).

Les standards sont accessibles depuis le site officiel : http://www.chiariglione.org/mpeg/.

6.1. La partie système

De part sa compatibilité avec le format MPEG1, le standard MPEG2 permet de construire des *flux de programmes* (PS). Un flux de programme est constitué de plusieurs *paquets de flux élémentaires* (PES) partageant une même horloge de synchronisation. La taille des paquets est toujours variable et relativement importante. La figure 6.1 illustre ces différents flux. Mais, dans le cadre de la télédistribution, MPEG2 construit aussi des *flux de transport* (TS). Un flux de transport comporte plusieurs programmes. Chaque programme a sa propre horloge de synchronisation et ses propres PES. Les erreurs et les pertes, habituellement rencontrées dans les réseaux, sont prises en compte avec un code correcteur de type CRC.

Il est possible de définir un flux de transport ne contenant qu'un seul programme. Il s'agit du mode *flux de transport à programme unique* (SPTS). Le mode SPTS est adapté aux télédiffusions telles que la *vidéo à la demande* (VOD) permettant la location et le téléchargement d'un programme (film, documentaire, information, etc.) pour une présentation sur un ordinateur ou un téléviseur *via* un réseau. Pour un enregistrement sur CD, le flux de programme est préféré, afin de rester compatible avec le standard MPEG1.

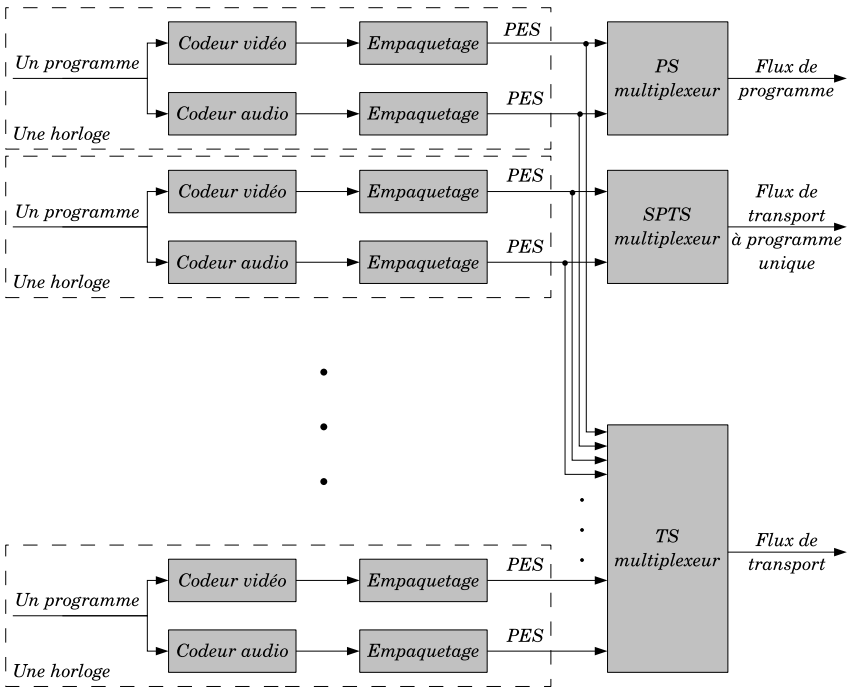


Figure 6.1. *Les différents flux générés par MPEG2*

6.1.1. *Le flux de transport*

Un flux de transport peut donc être constitué de PES, de PS, de SPTS ou d'autres TS. Les longueurs des paquets des flux de transport sont fixées à 188 octets. Le débit de chaque composant élémentaire (PES) peut être fixe ou variable. Ainsi, le débit binaire global est fixe ou variable.

Alors que le standard MPEG1 suppose l'enregistrement des flux de programme sans erreur, le standard MPEG2 distribue les flux sur des réseaux plus ou moins stables, plus ou moins sûrs. Les erreurs de diffusion rencontrées sont des modifications au niveau des bits ou des pertes, partielles ou totales, de paquets. Pour palier aux pertes, les en-têtes des paquets comportent un champ d'identification ordonnée. Les modifications intempestives de bits sont détectées par l'utilisation de codes correcteurs de type CRC.

Les en-têtes des paquets des flux de transport et des flux de programme comportent plusieurs champs. Le champ d'identification du paquet (PID[1]) est ordonné de façon à détecter toute perte de paquet. D'autres champs sont définis pour faciliter certaines opérations, comme :

1) extraire les paquets d'un programme du flux de transport, pour le décoder et le présenter ;

2) extraire les paquets d'un programme du flux de transport, pour construire un autre flux de transport à programme unique (SPTS) ;

3) extraire les paquets d'un ou de plusieurs programmes d'un ou de plusieurs flux de transport, pour construire un nouveau flux de transport ;

4) extraire les paquets d'un programme d'un flux de transport, pour construire un flux de programme (PS) ;

5) convertir un flux de programme (PS) en un flux de transport (SPTS) pour le diffuser sur un réseau. Les paquets sont redimensionnés (longueur fixe de 188 octets) et les codes correcteurs sont ajoutés.

La partie système du décodeur opère sur l'ensemble du flux de transport (*multiplex-wide operations*) ou sur des flux élémentaires séparés (*stream-specific operations*).

Un flux de transport comportant généralement plusieurs programmes, un jeu de tables est utilisé pour permettre la prévisualisation et la sélection des programmes par l'utilisateur final. Ce dernier utilise des guides électroniques de programmation (EPG[2]).

---

1. *Paquet IDentifier*.
2. *Electronic programming guide*.

Un EPG peut simplement permettre la sélection d'un programme, comme le fait le boîtier TV standard. Mais il peut aussi comporter un certain nombre de filtres sur les thèmes, les heures et les jours autorisés et bien d'autres critères encore.

Parmi ces tables, les trois plus importantes sont :

– PAT. La *table d'alocation des programmes*[3] contient les identifiants et les informations de l'ensemble des programmes d'un flux de transport. La PAT est régulièrement insérée dans le flux de transport toutes les 20 à 100 msec pour une mise à jour de la liste des programmes (voir figure 6.2) ;

– PMT. La *table d'association des programmes*[4] contient les identifiants et les informations complémentaires des ES audio et vidéo d'un programme. Les identifiants, de sigle PID, permettent de retrouver facilement les ES du programme (voir figure 6.2) ;

– CAT. La *table des accès conditionnels*[5] contient les informations nécessaires pour décoder les programmes cryptés.

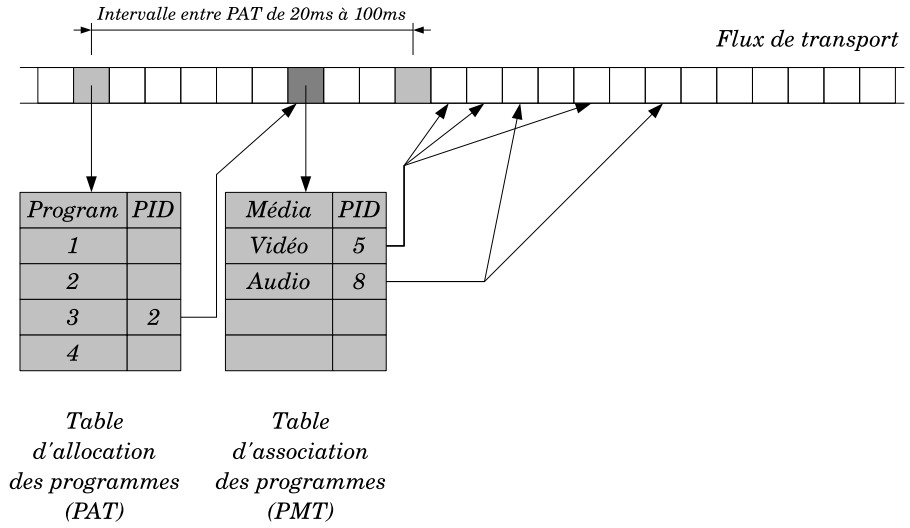


Figure 6.2. *Les tables du standard MPEG2 : le décodeur lit en premier la table PAT qui fournit les identifiants PID des tables PMT des programmes. Chaque table PMT fournit les identifiants PID des ES d'un même programme; si ce programme est crypté (VOD, chaînes payantes, etc.), une table CAT est ajoutée au flux de transport.*

3. *Program Allocation Table*.

4. *Program Map Table*.

5. *Conditional Access Table*.

6.1.2. *Le contrôle d'erreurs*

L'algoritme de contrôle des bits est celui utilisé par Ethernet. C'est un *contrôle de redondance cyclique* (CRC)[6], sur 32 bits. Le polynôme de validation vaut :

$$\begin{aligned} P(x) &= x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + \dots \\ & \quad x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \end{aligned}$$

Rappelons brièvement le principe. Pour ajouter un code de contrôle d'erreur à un message binaire M, le codeur le divise par le code binaire du polynôme $P(x)$. Le reste de la division est ajouté au message.

Le décodeur récupère le message complété de son code. Il effectue la division par le même polynôme. Si le message ne comporte pas d'erreur, le reste de la division est nul (voir annexe C.1).

6.2. La partie vidéo

L'algorithme général du standard MPEG1 est conservé (voir section 5.2). L'espace couleur par défaut est le *YCrCb*. Les I-images sont analysées en fréquences par une DCT dont les coefficients sont quantifiés avant d'être codés. Il y a toujours un module de compensation de mouvement pour le codage des P-images et des B-images.

Toutefois, la diffusion sur les canaux de la télévision standard[7] (SDTV) demande à s'adapter aux images *entrelacées*. Du fait de son étroite bande passante, la télévision standard découpe les *images* (*frames*) en *trames* (*fields*). Une trame est l'ensemble des lignes paires (resp. impaires) d'une image. Ainsi, une image est décrite par deux trames successives. La trame paire est parfois appelée *trame supérieure* (*top field*) et la trame impaire la *trame inférieure* (*bottom field*). La figure 6.3 illustre le mode progressif (à base d'images) et le mode entrelacé (à base de trames).

Il s'ensuit que la structure en macroblocs est différente suivant que la vidéo à coder est composée d'images ou de trames (voir figure 6.4) :

1) dans le cas d'images, la structure des macroblocs est identique à celle utilisée par le standard MPEG1 ;

2) dans le cas de trames, les macroblocs respectent la séparation en trames; ils sont constitués, soit de lignes paires, soit de lignes impaires.

---

6. *Cyclic Redundancy Check*.

7. La télévision standard se distingue de la *télévision haute définition* qui est à affichage progressif.

Les trames sont typées comme les images :

– les *I-trames* sont codées en mode *intra* (sans prédiction) ;

– les *P-trames* sont prédites à partir de trames de référence antérieures (suivant l'échelle temporelle de la vidéo source) ;

– les *B-trames* sont prédites à partir des trames de référence antérieures et postérieures.

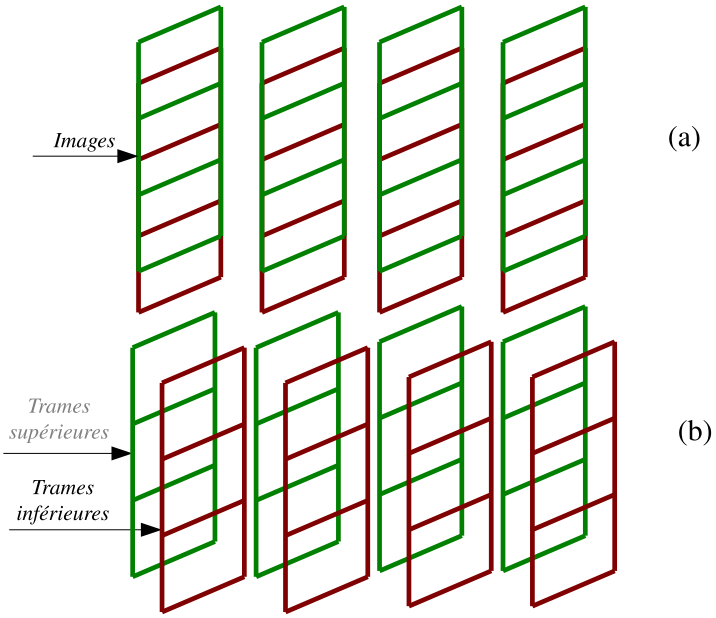


Figure 6.3. *Codage d'une vidéo (a) en mode progressif et (b) en mode entrelacé*

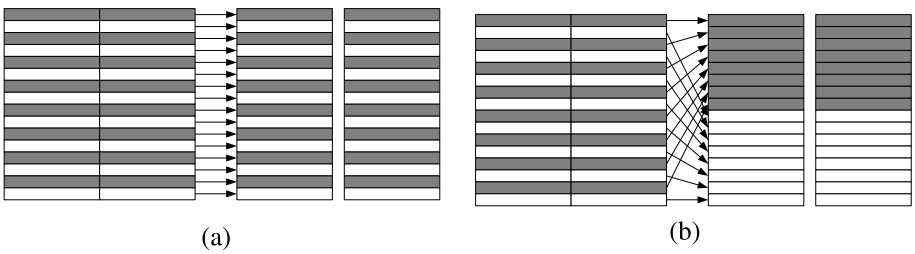


Figure 6.4. *Structure de la composante Y d'un macrobloc pour un codage DCT (a) en images, (b) en trames*

La taille des blocs traités par la DCT ne change pas : 8×8. En plus des macroblocs de taille 16×16, il est possible d'utiliser des macroblocs de taille 8×16 afin de s'adapter au mode entrelacé (vidéo en trames).

Le standard MPEG2 utilise les deux taux d'échantillonnage chromatique, 4:2:2 et 4:2:0, illustrés en figure 6.5. Le format chromatique 4:2:2 a le même facteur d'échantillonnage vertical pour les composantes *Cr* et *Cb* que pour la composante *Y*. En revanche, le facteur d'échantillonnage horizontal est divisé par deux. Ainsi, à deux sites chromatiques correspondent quatre sites de luminosité *Y*.

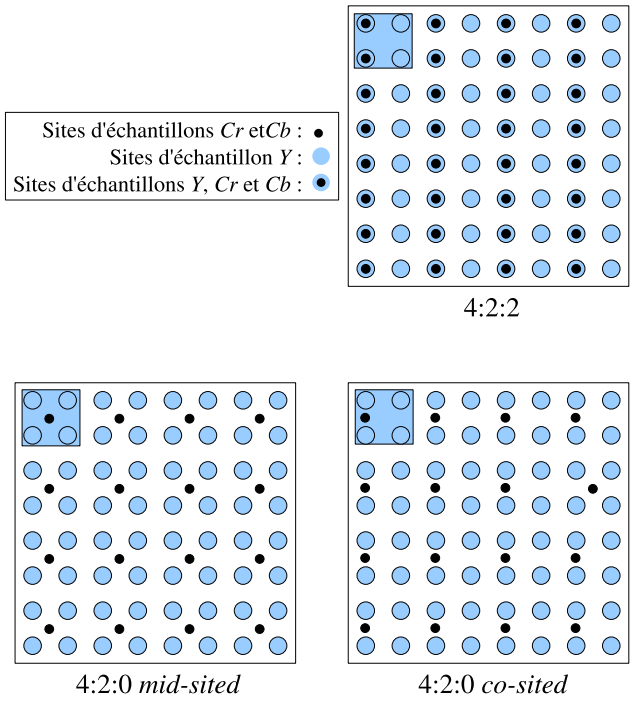


Figure 6.5. *Sous-échantillonnages 4:2:2 et 4:2:0*

Le format chromatique 4:2:0 divise par deux les facteurs d'échantillonnage en lignes et en colonnes. Un site chromatique est associé à quatre sites de luminosité. Ce format est dit *mid-sited* quand la valeur *Cr* (resp. *Cb*) est la moyenne des quatre valeurs chromatiques fournit par les sites initiaux; c'est-à-dire ceux qui ont conservé leurs valeurs de luminosité *Y*. Le format est dit *co-sited*, quand la valeur chromatique *Cr* (resp. *Cb*) est la moyenne des valeurs des deux sites de même colonne paire.

La structure des macroblocs associés à ces deux formats est montrée en figure 6.6. En mode progressif (à base d'images), le parcours en zigzag des coefficients AC quantifiés (des blocs de la DCT) se fait suivant l'algorithme déjà utilisé par le standard MPEG1. Pour le mode entrelacé, le parcours montré par la figure 6.7b est utilisé.

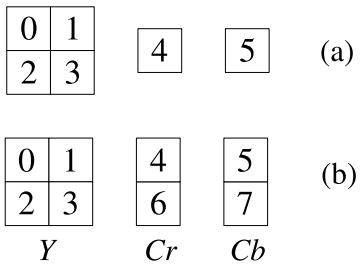


Figure 6.6. *Structure des macroblocs (a) du format chromatique 4:2:0 (*mid- *et* co-sited*) et (b) du format 4:2:2*

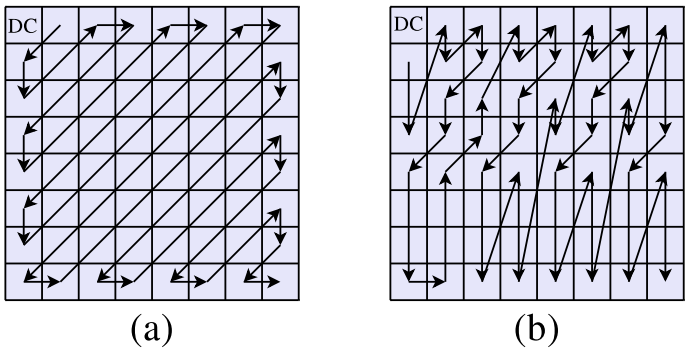


Figure 6.7. *Parcours en zigzag (a) classique (du format MPEG1) (b) en mode entrelacé*

6.2.1. *La compensation de mouvement*

La compensation de mouvement s'effectue en deux étapes. Pour chaque macrobloc à coder, une première étape de prédiction fournit les macroblocs de référence[8] et les vecteurs de déplacement correspondant. L'étape suivante construit le macrobloc résiduel provenant de la différence entre le macrobloc à coder et ses macroblocs de référence.

8. Dans certains cas, il n'y a qu'un seul macrobloc.

Il existe quatre modes de prédiction :

1) la *prédiction en image* est similaire à la prédiction du standard MPEG1 (voir paragraphe 5.2.2) ;

2) la *prédiction en trame*, détaillée au paragraphe 6.2.2, extrait les macroblocs, à coder et de référence, des trames au lieu de les sélectionner dans les images. Malgré cela, elle concerne aussi bien les trames (video entrelacée) que les images (vidéo progressive) ;

3) la *prédiction* 16×8 découpe chaque macrobloc 16×16 d'une trame en deux sous-macroblocs 16×8 (huit lignes, seize colonnes). Le paragraphe 6.2.3 détaille son fonctionnement ;

4) la *prédiction dual prime* est décrite au paragraphe 6.2.4.

6.2.2. *Prédiction en trames*

La trame de référence, de la prédiction en trames d'une P-trame, est soit la trame supérieure récemment codée (et décodée) soit la trame inférieure récemment codée (et décodée). Pour chaque macrobloc de cette P-trame à coder, l'estimation de mouvement fournit le macrobloc de référence ayant minimisé le critère de dissimilitude et le vecteur de déplacement correspondant. La figure 6.8 montre le cas général.

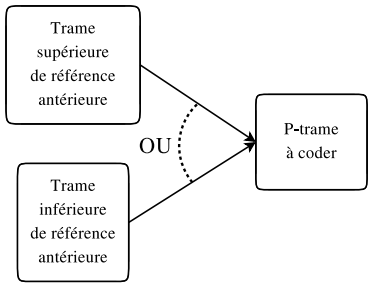


Figure 6.8. *Prédiction en trame d'une P-trame*

Pour une image I, si le codage des deux trames, T_1 et T_2, se fait suivant l'ordre indiqué par les indices des trames (T_1 avant T_2), alors les deux trames de référence pour le codage de T_1 proviennent de la même image de référence I'.

Il s'ensuit que le codage de T_2 fait référence à la trame T_1 (qui vient d'être codée). La deuxième trame de référence pour T_2 est choisie de même parité que T_2. La figure 6.9 montre le choix des trames de référence pour le codage de ces deux trames.

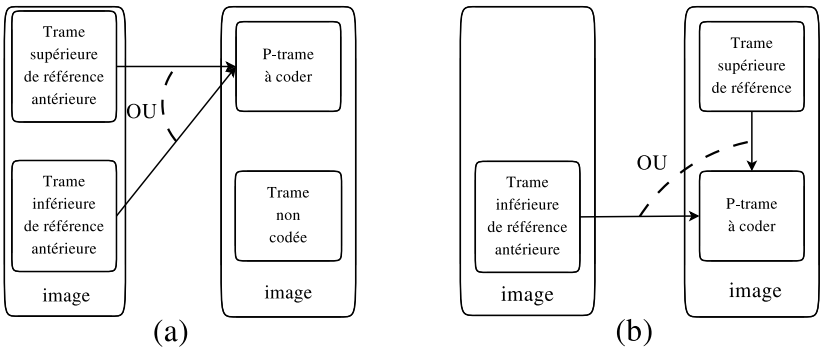


Figure 6.9. *(a) codage de la première trame T_1, (b) codage de la deuxième trame T_2*

6.2.2.1. *La prédiction en trames des P-images*

La prédiction en trames d'une P-image fournit deux vecteurs de déplacement : un vecteur pour chaque trame de la P-image. La P-image est observée comme étant un ensemble de deux trames à prédire.

Lors de la prédiction de la première trame de la P-image, les deux trames de référence proviennent de la même image de référence. Pour la prédiction de la deuxième trame, si celle-ci est une trame inférieure (resp. supérieure), le choix se fait entre la P-trame inférieure (resp. supérieure) d'une image de référence récemment codée et la P-trame supérieure (resp. inférieure) de la même image. La figure 6.10 illustre ces deux situations.

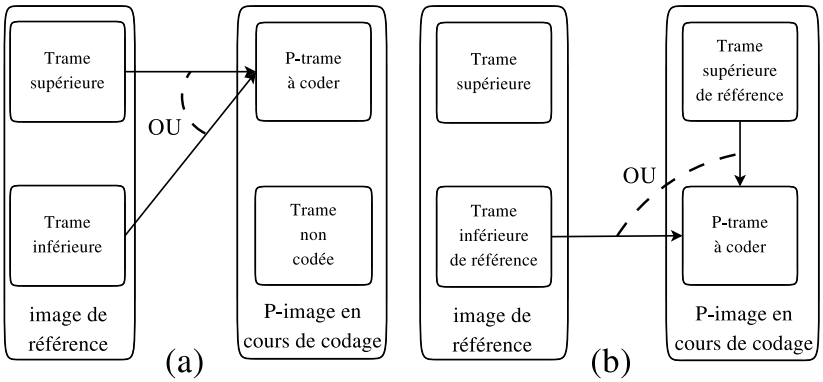


Figure 6.10. *Prédiction de la P-image : (a) le premier vecteur de déplacement est choisi parmi ceux provenant des deux trames de l'image de référence, (b) le deuxième vecteur de déplacement est choisi parmi ceux provenant soit de la trame de parité inverse dans la P-image soit de la trame de l'image de référence, de même parité.*

6.2.2.2. *La prédiction en trames des B-trames*

Pour une B-trame, la prédiction en trames est similaire au cas général de la prédiction en trame d'une P-trame, mais, les références proviennent d'une image de référence antérieure et d'une image de référence postérieure. La figure 6.11 illustre le procédé.

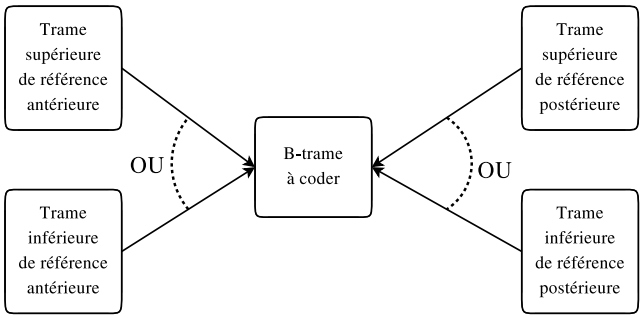


Figure 6.11. *Prédiction en trames d'une trame d'une B-image*

6.2.2.3. *La prédiction en trames des B-images*

La prédiction en trames d'une B-image fournit quatre vecteurs : deux vecteurs pour chaque trame de la B-image.

6.2.3. *La prédiction* 16×8

La prédiction d'un macrobloc 16×16 d'une P-trame fournit un vecteur de déplacement par sous-macrobloc 16×8 (voir figure 6.12a). La prédiction d'une B-trame fournit de deux à quatre vecteurs de déplacement associés à la trame de référence antérieure et à la trame de référence postérieure (voir figure 6.12b).

La prédiction d'une P-image fournit deux vecteurs de déplacement associés à l'image de référence :

– si le macrobloc appartient à la première trame de la P-image à être codée, le vecteur de déplacement de chacun de ses sous-macroblocs 16×8 correspond à un vecteur de déplacement associé à un sous-macrobloc, provenant soit de la trame supérieure soit de la trame inférieure de l'image de référence (voir figure 6.12c) ;

– si le macrobloc appartient à la deuxième trame de la P-image à être codée, le vecteur de déplacement de chacun de ses sous-macroblocs 16×8 correspond à un vecteur de déplacement associé soit à un sous-macrobloc de la première trame de la P-image soit à un sous-macrobloc d'une trame de référence antérieure de même parité que la trame prédite (voir figure 6.12d).

La prédiction d'une B-image délivre de deux à quatre vecteurs de déplacement.

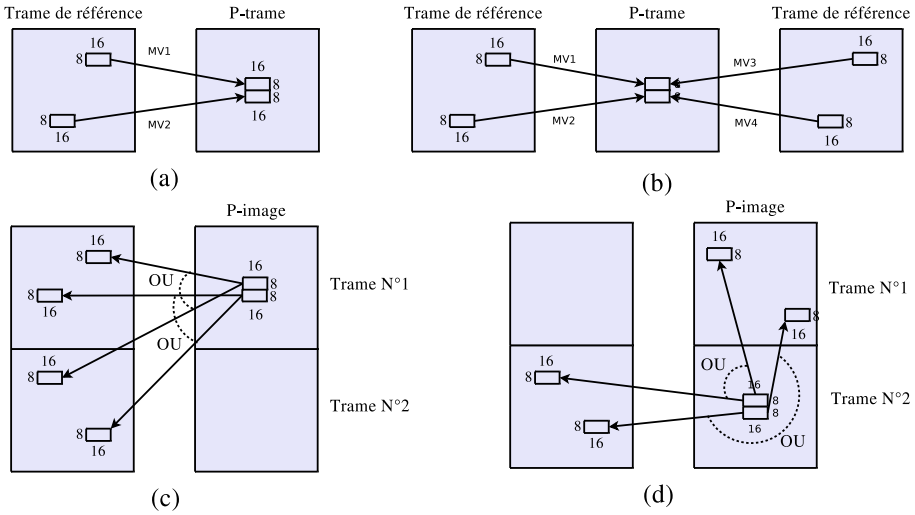


Figure 6.12. *Prédiction* 16×8 *: (a) pour une P-trame, (b) pour une B-trame, (c) pour la trame n°1 d'une P-image, (d) pour la trame n°2 d'une P-image*

6.2.4. *La prédiction* dual prime

La prédiction *dual prime* d'un macrobloc d'une P-trame transmet un vecteur, MV, et un vecteur différentiel, dmv, de faible amplitude (gain en taux de compression). Le vecteur MV représente le déplacement entre le macrobloc à coder (de la P-trame) et le macrobloc correspondant dans la trame de référence. Cette trame de référence est de même parité que la trame à coder (voir figure 6.13a).

Le vecteur différentiel dmv codifie la différence entre le vecteur MV et le vecteur de déplacement MV', associé à un deuxième macrobloc de référence. Ce macrobloc provient de la même image antérieure que celle utilisée pour le vecteur MV. Mais il se situe dans la trame de parité inverse. Le vecteur différentiel tient compte du décallage spatial entre les deux trames de référence et du décallage temporel dû aux dates de codage des deux trames. La figure 6.13b illustre le calcul du deuxième vecteur de déplacement MV', à l'aide du vecteur MV et du vecteur différentiel dmv.

Une fois les deux vecteurs de déplacement MV et MV' calculés, les deux macroblocs qui en découlent sont moyennés pour former la prédiction.

Cette technique s'étend naturellement aux P-images. La prédiction dual prime est appliquée à chacune des trames de la P-image à coder. La prédiction transmet alors deux vecteurs de déplacement $\mathrm{MV}_{\mathrm{sup}}$ et $\mathrm{MV}_{\mathrm{inf}}$, ainsi que deux vecteurs différentiels $\mathrm{dmv}_{\mathrm{sup}}$ et $\mathrm{dmv}_{\mathrm{inf}}$.

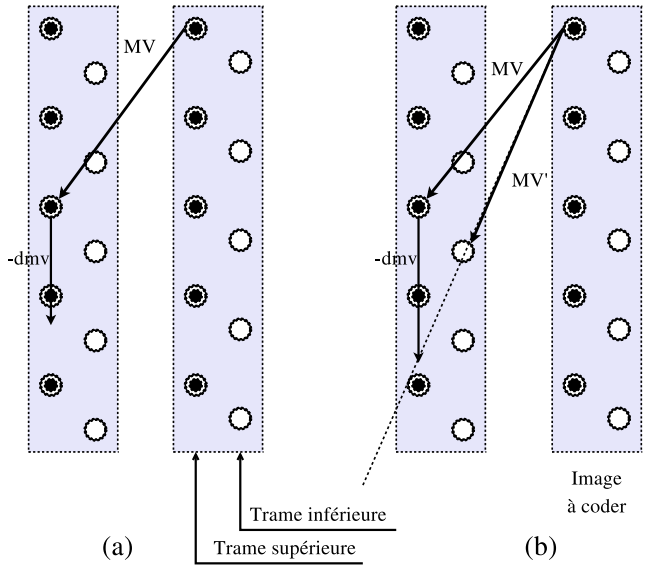


Figure 6.13. *Prédiction dual prime : (a) le vecteur* MV *indique le déplacement entre les macroblocs des trames de même parité ; (b) Le vecteur différentiel,* dmv*, permet de reconstruire le vecteur* MV′*, associé à la trame de parité inverse*

Ce mode simule une prédiction en B-images appliquée aux P-images mais avec les deux références antérieures.

6.2.5. *La granularité*

La *granularité*[9] découpe une vidéo source en plusieurs flux afin d'adapter le codage à différents débits binaires. Un *flux principal*, adapté aux faibles débits, permet au décodeur de fournir une présentation approximative de la vidéo source. Ensuite, si le débit binaire le permet, un ou deux autres flux peuvent être également envoyés au décodeur. Ces *flux comlémentaires* viennent améliorer la présentation de la vidéo. Le flux principal est voulu plus robuste aux perturbations du réseau que les flux complémentaires.

L'ensemble des flux forme une hiérarchie. La vidéo à sa plus basse résolution forme le flux principal. Les flux complémentaires s'imbriquent les uns dans les autres en apportant au fur et à mesure une amélioration à la vidéo basse résolution fournie par le flux principal.

9. Traduction libre pour *scalability*.

Lors de la compensation de mouvement, l'estimation de mouvement profite de cette *structure hiérarchique*. Une prédiction initiale est faite sur le flux principal. Ce flux étant une approximation de la source, la prédiction est grossière, mais, efficace en temps. Cette première prédiction est ensuite utilisée pour démarrer la prédiction sur le premier flux complémentaire. Le procédé est répété pour tous les flux complémentaires : la prédiction obtenue à partir du premier flux complémentaire sert d'initialisation pour la prédiction avec le deuxième flux complémentaire, etc.

Il existe cinq modes de granularité :

1) la granularité SNR. Le sigle SNR signifie *rapport signal sur bruit*[10] où le bruit désigne le bruit de quantification. La figure 6.14 illustre le codage en granularité SNR. Le codeur applique une forte quantification aux coefficients de la DCT pour former le flux principal. Lors du décodage, le *flux principal* fournit une version déformée de la vidéo originale. L'ensemble des déformations[11] est le *bruit de quantification* (voir chapitre 3 volume 1). Ce bruit de quantification est codé dans un *flux complémentaire* ;

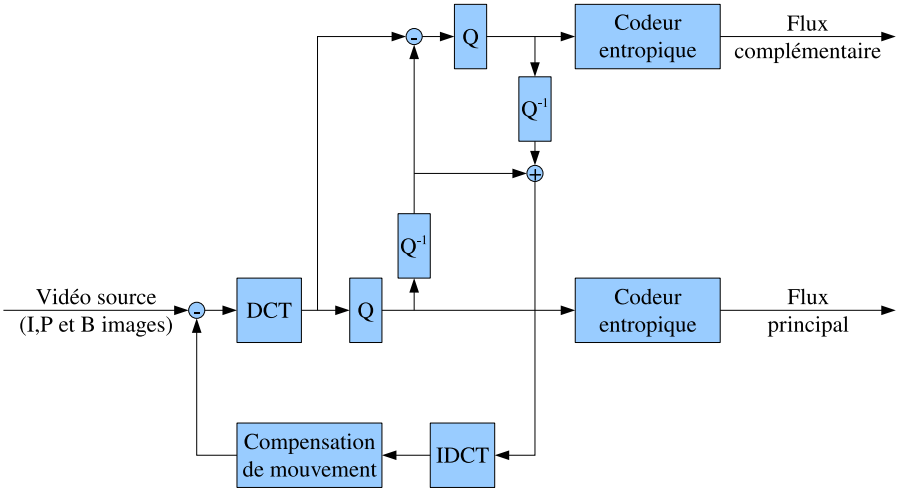


Figure 6.14. *Granularité SNR*

2) la granularité spatiale. Le schéma de codage en granularité spatiale est donné en figure 6.15. Le *flux principal* est construit à partir de la vidéo source sous-échantillonnée. Le sous-échantillonnage se fait par filtrage des hautes fréquences (voir

10. *Signal to Noise Ratio.*

11. Les différences entre les coefficients de la DCT et leurs versions quantifiées.

chapitre 3 volume 1). Classiquement, l'image est réduite de moitié. Les hautes fréquences, éliminées du flux principal, forment le *flux complémentaire*. Lors de la prédiction de mouvement, l'image basses fréquences – c'est-à-dire le *flux principal* – est utilisée pour initialiser le processus. Ensuite, les macroblocs prédits sont suréchantillonnés afin d'être mis en correspondance avec les macroblocs du flux complémentaire. Puis, la prédiction est affinée. Typiquement, le flux principal peut être utilisé pour un affichage au format SDTV. En ajoutant le flux complémentaire, la reconstruction de la vidéo répond aux critères du standard HDTV ;

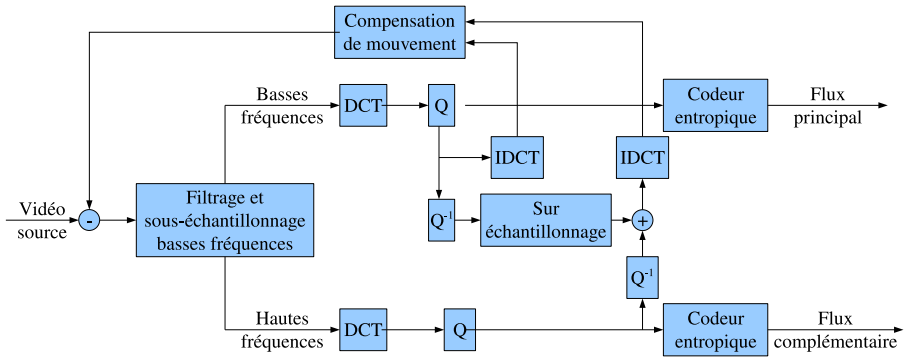


Figure 6.15. *Granularité spatiale*

3) la granularité temporelle. Le *flux complémentaire* et le *flux principal* sont obtenues en diminuant le taux de raffraichissement. Typiquement, le taux est diminué de moitié. Ainsi, le flux principal code une image sur deux. Le flux complémentaire code les images restantes (voir figure 6.16).

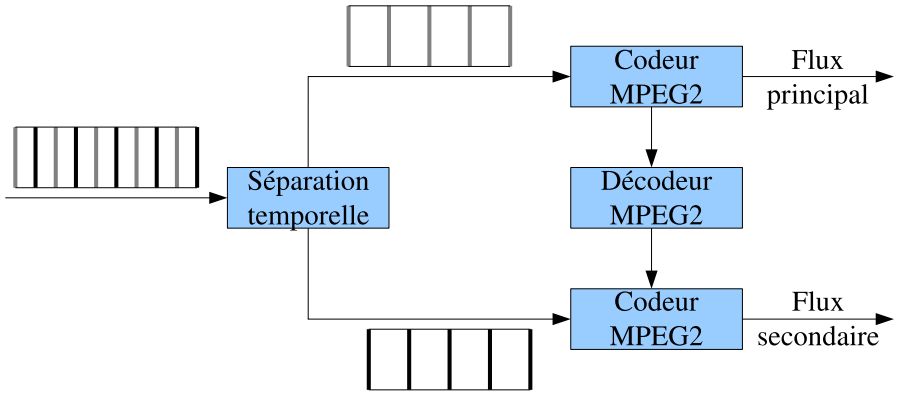


Figure 6.16. *Granularité temporelle*

La prédiction de mouvement utilise cette séparation temporelle. Le flux principal est codé classiquement. En revanche, les P-images et les B-images du flux complémentaire sont prédites à l'aide des images des deux flux :

a) une P-image du flux complémentaire est prédite à partir d'une I-image ou d'une P-image antérieure (dans le temps) provenant du même flux. Elle peut également être prédite à partir d'une I-image, d'une P-image ou d'une B-image du flux principal. L'image de référence, lorsqu'elle provient du flux principal, peut être antérieure ou postérieure à l'image à prédire (voir figures 6.17a, 6.17b et 6.17c). Cela signifie que le codage (resp. le décodage) du flux principal est en avance sur le codage (resp. le décodage) du flux complémentaire,

b) une B-image du flux complémentaire est prédite par couple d'images provenant des deux flux comme le montre les figures 6.17d, 6.17e et 6.17f ;

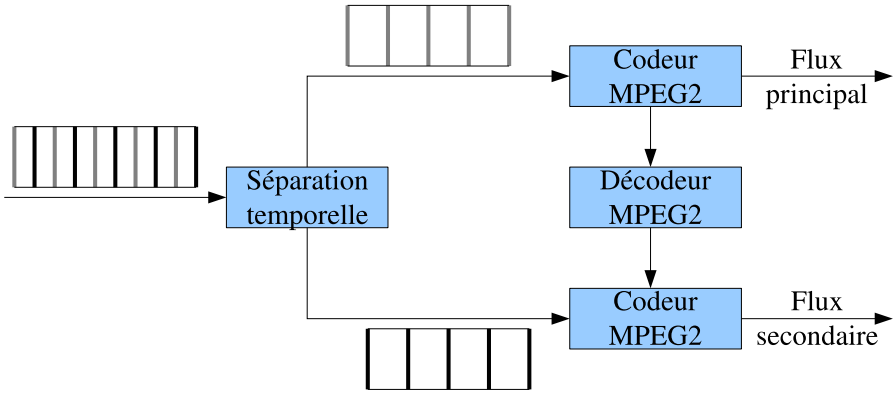


Figure 6.17. *La granularité temporelle : (a, b, c) d'une P-image, ou (d, e, f) d'une B-image*

4) la granularité hybride. Il s'agit de combiner deux parmi trois des granularités précédentes. Avec les options de granularité précédentes, la structure hiérarchique est composée d'un flux principal et de deux flux complémentaires. Par exemple, le flux principal est obtenu par granularité temporelle. A partir de la séquence d'images restante, la granularité SNR permet de construire les deux flux complémentaires ;

5) le regroupement. Les coefficients de la DCT sont séparés en deux groupes. Les coefficients des basses fréquences forment le flux principal tandis que ceux des hautes fréquences servent à la construction du flux complémentaire. Cette option est utile pour une transmission progressive sur des réseaux sensibles aux bruits.

REMARQUE 6.1.– *La granularité temporelle, fait intervenir des B-images du flux principal comme des images de référence pour la prédiction des P-images et des B-images du flux complémentaire.*

6.2.6. *Les niveaux et profils*

Les codeurs du standard MPEG2 sont des codeurs paramétrés par :

– des vidéos entrelacées ou progressives (constituées de trames ou d'images) ;

– des taux d'échantillonnages couleurs 4:2:0 ou 4:2:2 avec la définition des macroblocs adaptée comme le montre la figure 6.6 ;

– des tailles d'images de 352×288 à 1920×1152.

Certaines contraintes d'utilisation sont imposées et les configurations autorisées sont décrites par des couples (*niveau*, *profil*) fournis par le tableau 6.1.

Profil / Niveau	*Simple*	*Main*	*SNR*	*Spatial*	*High*
High		OK			OK
High-1440		OK		OK	OK
Main	OK	OK	OK		OK
Low		OK	OK		

Tableau 6.1. *Couples (niveau, profil) autorisés par le standard MPEG2*

6.2.6.1. *Les niveaux*

Les niveaux (*levels*) définissent les tailles d'images et les taux de raffraîchissement :

	Low	*Main*	*High-1440*	*High*
Taille del'image	325×288	720×576	$1\,440 \times 1\,152$	$1\,920 \times 1\,152$
Taux de raffraîchissement	30	30	60	60

Les niveaux *High-1440* et *High* correspondent aux résolutions hautes utilisées principalement pour la production. Le niveau *High* correspond au format 16:9.

6.2.6.2. *Les profils*

Les profils (*profils*) sont définis par les facteurs d'échantillonnage et les types d'images (I, P et B) utilisées (voir tableau 6.2) :

1) le profil *Simple* ne prend pas en considération les B-images. Ceci simplifie le codeur et le décodeur. Seul le niveau, *Main*, est autorisé avec ce profil ;

2) le profil *Main* est défini pour tous les niveaux. Dans la majorité des applications de télédiffusion en SDTV, ce profil est utilisé conjointement avec le niveau *Main*; le sigle MP@ML est utilisé pour l'indiquer : *Main Profil at Main Level* ;

3) les profils SNR, *Spatial* et *High* proposent une codification *graduelle* (voir tableau 6.2) :

a) le profil SNR supporte les niveaux *Low* et *Main* ;

b) le profil *Spatial*, le niveau *High* ;

c) le profil *High*, tous les niveaux, excepté le niveau *Low*.

	Simple	*Main*	*SNR*	*Spatial*	*High*
Facteur d'échantillonnage	4:2:0	4:2:0	4:2:0	4:2:0	4:2:0 ou 4:2:2
Type d'images	I, P	I, P, B	I, P, B	I,P, B	I, P, B
Granularité	Non	Non	SNR	SNR ou spatial	SNR ou spatial

Tableau 6.2. *Caractéristiques principales des profils*

Ces notions de profils et de niveaux permettent d'adapter le débit à la demande (stockage sur DVD, télédiffusion). Elles sont reprises et étendues par le standard MPEG4 pour de nouveaux supports réseaux comme la téléphonie mobile.

6.3. La partie audio

La partie audio fonctionne en mode BC (*Bakward Compatible*) ou en mode AAC (*Advenced Audio Compression*).

6.3.1. *Le mode BC*

Le mode BC intègre les trois couches définies par le standard MPEG1. Pour diminuer le débit binaire (< 64kbit/s par canal), trois fréquences d'échantillonnage sont ajoutées : 16 kHz, 22,05 kHz et 24 kHz. Les fenêtres d'observation du signal, l'allocation de bits et le modèle psychoacoustique des couches I, II et III sont alors adaptés à ces nouvelles fréquences.

A ceci s'ajoute cinq nouveaux canaux, plus un canal spécifique. L'ensemble de ces canaux constituent le format 5.1 du *home-cinéma*. Les cinq premiers canaux sont dédiés aux hauts-parleurs avant gauche (*left front* – L), avant central (C), avant droit (*right front* – R), arrière gauche (*left surround* – LS) et arrière droite (*right surround* – RS). Le canal spécifique LFE (*Low Frequency Enhancement*) restitue les basses fréquences. C'est le caisson des basses placé généralement proche des auditeurs (voir figure 6.18).

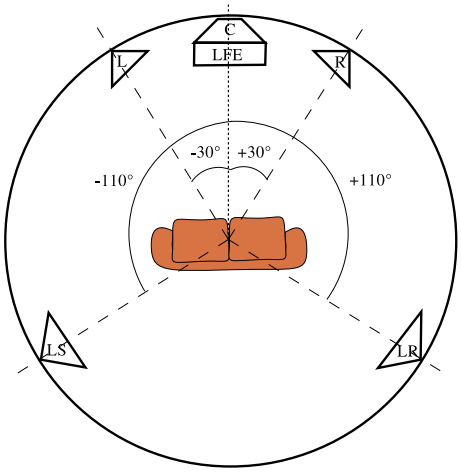


Figure 6.18. *Configuration spatiale du home-cinéma 5.1*

Le codage se fait donc sur ces 5+1 canaux avec la possibilité de mixer leurs signaux pour reconstruire un signal stéréophonique. Ainsi, la compatibilité avec des matériels ne supportant que le format MPEG1 (stéréophonique) est assurée.

Pour des applications telles que la HDTV, des commentaires multilingues peuvent être ajoutés. Par exemple, les 5+1 canaux peuvent être utilisés pour une écoute stéréophonique avec le canal C dédié aux commentaires pour malvoyants ou aux commentaires d'émissions sportives.

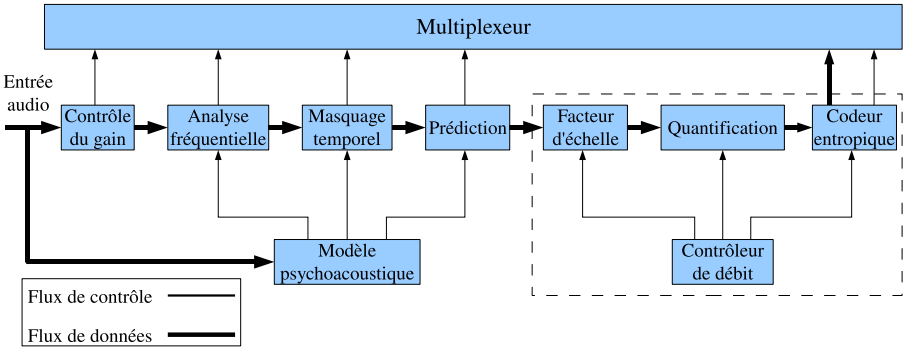


Figure 6.19. *Le codeur du mode AAC*

6.3.2. *Le mode AAC*

Le mode AAC a été industriellement développé par Apple pour ses iPhone et iPod. Les fichiers audio Apple ont pour suffixe *.m4u*. Ce mode permet de diminuer le débit pour une qualité audio identique à celle du codage en mode BC. La figure 6.19 donne le fonctionnement général du mode AAC :

1) le module *contrôle du gain* découpe le signal en entrée en quatre sous-bandes de 6 kHz de large, à l'aide d'un banc de filtres. Au sein de chaque sous-bande, les brusques variations des coefficients des fréquences sont détectées. Lorsqu'il y a une forte variation, sa date d'arrivée est enregistrée dans le flux audio. Les coefficients correspondant à ces variations sont corrigés pour une meilleure compression ;

2) l'*analyse fréquentielle* est faite avec une MDCT pure (sans un banc de filtres la précédant). Les échantillons sont regroupés en blocs longs et courts avec un recouvrement de 50 %. Les blocs longs, de taille 2 048, délivrent 1 024 lignes (ou sous-bandes) fréquentielles. Les blocs courts, de taille 256, délivrent 128 lignes fréquentielles. Les blocs longs sont adéquats pour l'analyse des basses fréquences, alors que les blocs courts le sont pour les hautes fréquences. Une préanalyse des transitions musicales permet d'estimer au mieux la taille du bloc à analyser. Le passage d'un bloc long à un bloc court, ou inversement, se fait progressivement comme pour le standard MP3 ;

3) le *modèle psychoacoustique* est similaire à celui du mode BC (c'est-à-dire à celui du standard MPEG1) ;

4) le *masquage temporel* (TNS[12]) est un algorithme complexe qui, au sein de chaque bloc, minimise les effets de dispersion temporelle (échos et prééchos) dus à l'analyse fréquentielle ;

5) la *prédiction* réduit les redondances dans les zones stationnaires du signal. Elle est adaptative et fonctionne par retour arrière; les blocs déjà traités servent d'estimateur aux paramètres de la prédiction ;

6) le module d'estimation du *facteur d'échelle*, δ, correspond au gain à appliquer aux coefficients spectraux lors de la quantification. L'ensemble des lignes fréquentielles est découpé en quatre bandes. Le gain est estimé pour chacune des bandes. Le facteur d'échelle de la première bande est estimé grâce au modèle psychoacoustique. Le facteur d'une bande supérieure augmente de 1,5 dB la valeur du facteur de la bande précédante ;

7) le module de quantification est non uniforme suivant la formule :

$$Q(x) = \text{sign}(x)\,\text{arrondi}\left(\left(\frac{x}{2^{\delta/4}}\right)^{3/4} + 0.4054\right)$$

où δ le facteur d'échelle qui définit le pas de quantification : $2^{\delta/4}$;

12. *Temporal Noise Shaping*.

8) le module de codage entropique est celui de Huffman. Il est appliqué aux coefficients quantifiés des sous-bandes et aux valeurs différentielles, $(\delta_i - \delta_{i-1})$, des facteurs d'échelle.

Le mode AAC supporte trois profils :

1) le profil de *faible complexité* (LC : *Low Complexity*) est le plus simple. Le module de prédiction est ignoré. Le masquage temporel est limité à l'ordre 12. C'est le profil le plus couramment utilisé ;

2) le profil *principal* (*main*) offre le meilleur rapport qualité/compression. Mis à part, le module de contrôle de gain, tous les modules sont utilisés ;

3) le profil à *taux d'échantillonnage graduel* (SSR : *Scalable Sampling Rate*) est utilisé pour des débits variables. Le module de prédiction est ignoré. En revanche, le module de contrôle de gain est utilisé.

6.4. La partie DSM-CC

La partie *contrôle et commande des médias numériques* (DSM-CC) est un ensemble de protocoles et de services *publics* pour le contrôle des flux MPEG et l'interaction avec ceux-ci. La référence de cette section est la synthèse proposée par le site Internet officiel du format MPEG : http://www.chiariglione.org/mpeg/tutorials/.

L'objectif initial est de fournir les commandes usuelles des cassettes vidéos VCR : avance rapide, retour arrière, pause. Puis, d'autres services ont été ajoutées : vidéo à la demande (VOD), téléachats, etc. Pour ce faire, la DSM-CC offre une indépendance des services par rapport à la couche transport. Divers réseaux – fibres optiques, coaxiaux, etc. – peuvent être utilisés sans discernement.

La partie DSM propose un ensemble de protocoles couvrant :

– le contrôle des sessions et des ressources réseaux ;

– la configuration d'un client ;

– le téléchargement vers un client ;

– le contrôle de type VCR du flux vidéo ;

– les services génériques pour les applications interactives ;

– les services génériques pour les applications de télédiffusion – carrousel d'objets.

La souplesse de la partie DSM-CC est un de ses atoûts. Chaque protocole peut être utilisé indépendamment ou conjointement avec d'autres protocoles, suivant les besoins des applications courantes. Le paragraphe qui suit présente le modèle de référence. Puis, les paragraphes suivants décrivent succinctement chacun de ces protocoles.

6.4.1. *Le modèle de référence*

La figure 6.20 montre le modèle relativement simple servant de référence à la DSM-CC. Le principe mis en œuvre est celui du *client/serveur*. Le client est habituellement un boîtier TV (*set-top box*) délivrant des produits multimédias. Le serveur founit les produits multimédias et les services correspondants. Il peut être distribué et donc avoir plusieurs nœuds de connexion au réseau.

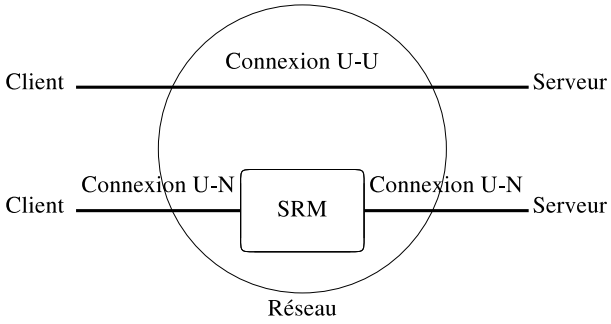


Figure 6.20. *Mode de fonctionnement*

Les définitions de réseaux et de connexions sont très générales pour que la DMS-CC soit utilisable avec une large collection de types de réseaux (fibres optiques, coaxiaux, fils torsadés, etc.). Les connexions peuvent être *point-à-point* (*peer-to-peer*) ou multipoints. Deux modes de communication sont possibles où le client et le serveur sont vus comme des *utilisateurs* (*users*) :

– *mode utilisateur-utilisateu*r. Le mode U-U (*User-User*) autorise un accès direct aux objets du serveur par le client. Les services accessibles sont installés chez le client par le serveur. Le fonctionnement repose sur le principe des *appels de procédures distantes* (RPC[13]). Le client peut aussi *interagir* en déposant des objets sur le serveur quand ce dernier l'autorise *via* les services proposés ;

– *mode utilisateur-réseau*. La partie DSM-CC est alors composée de trois éléments : le client, le serveur et le *gestionnaire des ressources et des sessions* (SRM[14]). Le serveur fournit les contenus audiovisuels au client *via* le SRM. Ce dernier alloue, relâche et contrôle les sessions et les ressources du (des) réseau(x). Le mode U-N (*User-Network*) définit un en-tête standard pour tous les messages U-N, afin de détecter et de supprimer ceux qui sont défectueux.

13. *Remore Procedure Call*.

14. *Session and ressource manager*.

6.4.2. *Le contrôle des sessions et des ressources réseaux*

La session est à la base du fonctionnement de la partie DSM-CC. Elle définit une interopérabilité entre deux utilisateurs. Elle permet de regrouper toutes les ressources nécessaires lors de l'activation d'un service. Par exemple, un client accède à un service d'achats en ligne, par la mise en place d'une session avec un serveur. Quand le client n'a plus besoin du service proposé par le serveur, la session est arrêtée.

Chaque session possède un identificateur de session, *sessionId*, utilisable sur tout le réseau. Ainsi, les ressources sont attribuées à une session en leur assignant le sessionId de la session. Ceci permet de compter les ressources utilisées par une session et de les gérer jusqu'à ce que la session soit arrêtée. Les ressources ne sont pas forcément gratuites et l'identificateur sessionId permet également de facturer les services rendus. Le protocole de démarrage d'une session permet d'authentifier le client et de vérifier s'il accepte la mise en place de cette session.

Lors d'une session, le serveur peut vouloir ajouter des ressources ou en supprimer, suivant l'état du service en cours. Il utilise alors un ensemble de messages *AddResource* (resp. *DeleteResource*) où les ressources sont décrites à l'aide de descripteurs de ressources (*Resource Descriptors*).

Les flux d'informations de connexion entre utilisateurs sont les principales ressources d'une session. Une session possède généralement plus d'une connexion utilisateurs. Typiquement, une session comprend un flux de contrôle pour les RPC et un flux de transport MPEG2. Une session plus sophistiquée utilise souvent d'autres flux pour, par exemple, séparer le graphisme d'arrière-plan du flux vidéo proprement dit.

Comme les contenus utilisés par le client peuvent provenir de différentes sources, celles-ci ne sont pas tenues de cloturer la session en un même point d'accès réseau.

Une ressource réseau peut être n'importe quel moyen de transport. Aussi, le SRM assure l'indépendance des services vis-à-vis des protocoles réseaux utilisés par les ressources. Certaines ressources établissent des connexions bout-à-bout entre le client et le serveur. Mais, une session peut regrouper plusieurs ressources dans une connexion multipoint. Par exemple, comme le montre la figure 6.21, le serveur peut être connecté à un réseau en fibre optique, alors que le client est raccordé à un réseau de type ADSL.

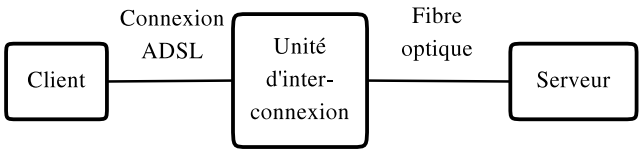


Figure 6.21. *Une connexion multipoint*

La DSM-CC définit un format de descripteur de ressources commun aux différents types de réseaux. Un descripteur de ressources est une structure de données transmise entre les utilisateurs et le réseau *via* des messages d'allocation de ressources. Il contient les valeurs appropriées au type de réseau (adresses du client et du serveur, table d'association des programmes – PMT – etc.). Il indique si la ressource est négociable et qui l'initie (le client, le serveur ou le réseau). Il fournit également un marqueur d'association qui permet de savoir à quelle connexion (couple client/serveur) la ressource correspond. Ainsi, toutes les ressources d'une même connexion ont le même marqueur. Il est de la responsabilité du SRM de regénérer des descripteurs lorsque le réseau change de configuration. Le descripteur informe sur l'utilisateur (client ou serveur) associé à la ressource.

Bien qu'une session soit souvent initiée par le client, dans certains cas, le serveur (ou un autre client) a besoin d'informer celui-ci qu'il devrait démarrer une session. Un message *PassThruReceipt* est envoyé au client qui peut alors accepter ou décliner l'offre.

6.4.3. *La configuration d'un client*

Dès le démarrage de la session, le client peut vouloir décrire sa configuration au réseau. Il utilise pour cela des messages *UNConfig* où le réseau en question est identifié. La configuration U-N peut être initiée par l'utilisateur (messages *UNConfigRequest* et *UNConfigConfirm*), par le réseau (messages *UNConfigIndication*, *UNConfigResponse*) ou suite à une écoute de l'utilisateur d'un canal de large diffusion (message *UNConfigIndication*).

Une fois la configuration terminée, l'utilisateur est informé des paramètres réseau à utiliser pour communiquer avec le SRM.

6.4.4. *Le téléchargement vers un client*

Le protocole de téléchargement est léger et rapide, afin de faciliter le transfert des données et des logiciels du serveur vers le client. Une fois la session démarrée, la DSM-CC permet au serveur de télécharger chez le client tout un environnement de travail. A l'initialisation du téléchargement, les périphériques du client, requis par le serveur, se décrivent *via* des descripteurs génériques. Le téléchargement peut être, soit en mode interactif avec un contrôle du flux, soit en mode par diffusion sans contrôle du flux.

En mode interactif, le téléchargement de données chez le client est suivi d'un message de confirmation de celui-ci.

Un serveur utilise le mode par diffusion pour télécharger les données sur plusieurs clients simultanément. Dans ce cas, les descripteurs de compatibilité, définis dans le message de démarrage du téléchargement, dirige le client vers le canal de large diffusion adéquat, avec le débit approprié.

Le fait de télécharger les logiciels chez le client supprime la vérification des versions de ceux-ci ainsi que les mises à jour qui doivent être régulièrement faites, lorsque les logiciels sont installés chez le client. Ceci permet de concevoir des boîtiers TV simplifiés et facilite les opérations de maintenance qui sont à la charge des serveurs.

6.4.5. *Le contrôle de type VCR du flux vidéo*

Les estampilles – DTS et PTS – d'un flux de transport MPEG n'informent pas sur le déroulement de la vidéo. Elles concernent uniquement les synchronisations entre les flux vidéo et les flux audio lors du décodage. Mais, la lecture d'une vidéo demande plus : positionnement aléatoire, lecture et retour arrière à vitesses variables, etc. Aussi, les interfaces utilisent une fréquence d'horloge propre à la partie DSM-CC, appelée *fréquence normale de lecture* (NPT[15]). Par défaut, cette fréquence donne le temps permettant une visualisation à vitesse normale du programme téléchargé. Quand le client opère une avance rapide (resp. un retour arrière), la fréquence d'horloge est augmentée (resp. diminuée). Le taux d'augmentation (resp. de diminution) de la fréquence définit la vitesse d'avance (resp. de retour arrière).

6.4.6. *Les services génériques pour les applications interactives*

En mode U-U, la DSM-CC fournit un ensemble d'interfaces génériques modulaires utilisables par une grande variété d'applications tels que les VOD, le téléachat ou, encore, l'enseignement à distance.

Ces interfaces sont écrites à l'aide d'un langage approprié, IDL[16]. Ce langage permet la création de deux types d'interfaces (voir figure 6.22) :

1) une *interface de portabilité d'application* (API[17]) permet aux programmeurs de construire des applications clients sans se soucier des problèmes de transport ;

2) une *interface d'interopérabilité de services* (SII[18]) permet la communication entre des clients et des serveurs, de différents constructeurs. Une SII définit un format binaire correspondant aux RPC choisies.

---

15. *Normal Play Time*.
16. *Interface Definition Language*.
17. *Application Portability Interface*.
18. *Service Interoperability Interface*.

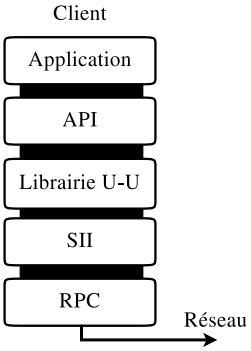


Figure 6.22. *API et SII*

6.4.7. *Les services génériques pour les applications de télédiffusion*

Lors d'une télédifusion, les téléchargements vers les clients se font en mode par diffusion (*broadcast*) qui n'opère pas de contrôle du flux. Autrement dit, un client (*via* un boîtier TV) doit pouvoir accéder à un programme alors que le serveur a déjà démarré le transfert vers ses clients.

Pour ce faire, le serveur découpe, au préalable, l'ensemble des données à diffuser en modules. Chaque module comporte un descripteur permettant de l'identifier. Le serveur envoie périodiquement les modules. Le client doit alors attendre d'avoir récupéré tous les modules pour reconstruire l'ensemble initial. Cet algorithme, appelé *Carrousel*, peut être de deux types :

1) le *carrousel de données* n'informe pas sur le contenu des modules. C'est au client d'arranger les modules dans des structures qui ont un sens pour lui ;

2) le *carrousel d'objets* est adapté aux situations complexes. Il s'agit d'un carrousel de données muni d'un système de fichiers standard. Les données sont munies de leurs types (fichiers, répertoires, etc.) et de leurs positions dans l'arborescence du système de fichier. Chaque module contient des fichiers et des répertoires regroupés de façon à optimiser l'espace mémoire qui est limité à 64 Ko. Le serveur envoie périodiquement les modules. Le client récupère les modules qui l'intéressent et reconstruit l'arborescence.

EXEMPLE 6.1.– *Soit un serveur qui diffuse l'arborescence décrite par la figure 6.23. Les deux premiers fichiers (*`index.html` *et* `image.jpg`*) sont regroupés dans un même module. Mais, le troisième fichier (*`mini_clip.mpeg`*) ne peut pas y être ajouté, car la limite en capacité (64 Ko) serait dépassée. En revanche, le fichier* `audio/mini_clip.aiff` *du répertoire* `audio` *peut être ajouté, ainsi que l'entrée au répertoire* `audio`*. Bien que dépassant la capacité d'un module, le fichier* `mini_clip.mpeg` *est stocké dans un module, car il n'y a pas d'autre solution. Il*

reste le répertoire `docs` *qui occupe plusieurs modules. Un module contient l'entrée au répertoire et un maximum de fichiers possibles dans la limite de la capacité du module : l'entrée* `docs` *et les deux fichiers,* `docs/Main.html` *et* `docs/Other.html`*. Le dernier module contient le fichier* `docs/Big.jpg`*. L'ordre d'envoi des modules dépend de l'utilisation faite des fichiers par le client. Comme le montre la figure 6.24, le module 1 est envoyé plus régulièrement que les autres.*

```
index.html              1048 octets
image.jpg               3240 octets
mini_clip.mpeg          130304 octets

audio                   <directory>
audio/miniclip.aiff     32830 octets

docs                    <directory>
docs/Main.html          27420 octets
docs/Big.jpg            59502 octets
docs/Option.html        26874 octets
```

Figure 6.23. *Exemple d'arborescence d'un serveur*

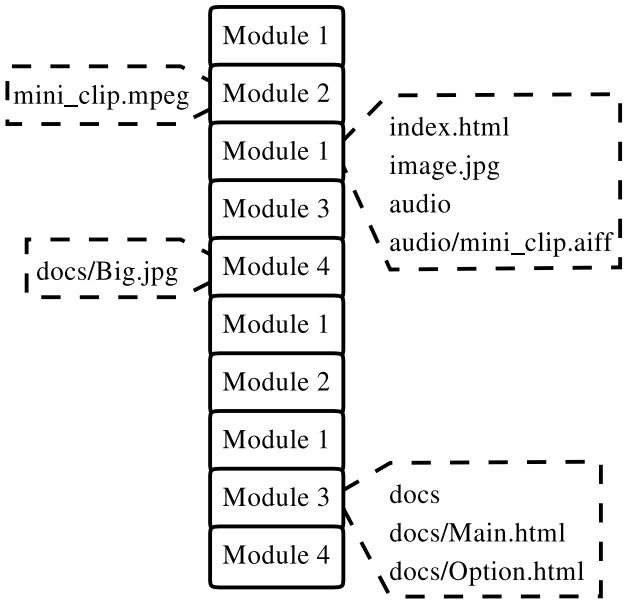


Figure 6.24. *Ordre d'envoi des modules*

6.5. Synthèse

MPEG2 délivre une description hiérarchique de l'audiovisuel qui augmente les taux de compression offert par MPEG1. Plusieurs flux hiérarchiquement interdépendants décrivent la vidéo. Cette description autorise également une plus grande souplesse dans le flux binaire fourni.

Cette *granularité* permet de s'adapter aux réseaux utilisés pour la télédiffusion. Ainsi, sur un réseau à faible débit (par exemple, la téléphonie mobile) un flux principal est envoyé. Ce flux offre une version légèrement dégradée de l'audiovisuel. Il est généralement robuste et muni de codes de contrôle d'erreurs pour la transmission.

Si le débit du réseau le permet, un flux secondaire s'ajoute au flux principal. Ce flux secondaire fournit les informations visuelles complémentaires qui augmentent la qualité de présentation de l'audiovisuel.

Ainsi, le format MPEG2 autorise un affichage paramétrable suivant la résolution spatiale de l'image et le taux de raffraîchissement (nombre d'images par seconde) de l'audiovisuel. Toutefois, les valeurs prises par ces paramètres sont contrôlées.

Un ensemble de niveaux de résolutions est défini. Les outils associés sont regroupés en profils. Les paramètres audiovisuels autorisés sont décrits par des couples de la forme (niveau, profil).

L'un des couples les plus connus est le fameux *MP@ML* qui associe le profil *Main* avec le niveau *Main*, utilisé par les lecteurs vidéos grand public.

L'intégration de la télédiffusion par le format MPEG2 a conduit à la création de nouveaux produits : conférences et enseignement à distance, téléachat, VOD, etc. Il a fallu intégrer ces nouvelles options de la manière la plus transparente qui soit. En particulier, lorsqu'un consommateur utilise son boîtier TV pour effectuer un téléachat ou commander un audiovisuel en ligne, tous les moyens réseaux disponibles doivent être utilisés sans que le consommateur ait à négocier et/ou à gérer quoi que ce soit.

Par exemple, lors d'une commande d'une vidéo en ligne, le client va simplement ouvrir une session qui lui permettra d'accéder au catalogue en ligne proposé par un serveur. Le réseau du client peut être un réseau sans fil et celui du serveur un réseau à fibres optiques. La transparence entre ces deux réseaux est de rigueur.

Pour répondre à ce besoin de transparence, une nouvelle partie est ajoutée au standard MPEG2 : la partie *commandes et contrôles des médias numériques* (DSM-CC). Cette partie autorise un client et un serveur à communiquer directement *via* des *appels de procédures distantes* (RPC) ou bien à communiquer par l'intermédiaire d'un *gestionnaire des sessions et des ressources* (SRM).

Si le serveur est installé sur un site unique, le gestionnaire est localisé sur le serveur. En revanche, si les ressources associées au serveur sont situées en différents lieux, le gestionnaire est multipoint.

Le DSM-CC utilise un *carrousel* d'objets pour la télédiffusion. Ce carrousel découpe les données et les logiciels proposés par un serveur en modules pour les diffuser régulièrement sur son réseau. Ainsi, lorsqu'un client fait appel à un service, il n'est pas utile de réinitialiser le téléchargement. Le client consulte les modules de ce carrousel pour y retrouver les données et les logiciels qui l'intéressent.

Chapitre 7

MPEG 4

Le standard MPEG4 apporte une évolution considérable dans la façon de modéliser un audiovisuel et de concevoir sa compression. Ses avancées autorisent MPEG4 à assembler des données naturelles avec des données synthétiques.

Certains pans de cette évolution sont encore (en 2008) sujets à des études théoriques ou de développement. En particulier, s'agissant du codage de la vidéo naturelle, nombre de tâches[1] sont encore du domaine de la recherche.

MPEG4 se décompose en parties comme ses prédécesseurs. Toutefois, il en compte plus d'une vingtaine. Les principales concernent :

– *part 1*. Le système qui assure la synchronisation et le multiplexage des flux vidéo et audio. Mais aussi, il assure une interactivité à tous les niveaux : du codeur au décodeur ;

– *part 2*. Le codage de la vidéo ;

– *part 3*. Le codage de l'audio ;

– *part 6*. Une interface, appelée *Delivery Multimedia Integration Framework* (DMIF), qui permet de développer une application sans se soucier des contraintes imposées par les différents réseaux rencontrés;

– *part 10* : un codec pour la vidéo, appelé *Advanced Video Coding* (AVC) ;

– *part 16*. Le codage des données visuelles synthétiques, appelé *Animation Framework eXtention* (AFX).

1. Par exemple, la segmentation d'images et de vidéos en régions significatives.

MPEG4 permet donc le codage fusionnel de données naturelles et synthétiques.

DÉFINITION 7.1.– *Les termes de données naturelles ou synthtiques, sonores ou visuelles correspondent à :*

1) une donnée naturelle visuelle est une photographie, un audiovisuel, etc. ;

2) une donnée naturelle sonore est un enregistrement d'un débat ou d'une composition musicale, etc. ;

3) une donnée synthétique sonore est le résultat d'une transcription automatique d'un texte en parole (TTS : Text-to-Speech*) ou encore une composition par ordinateur d'une œuvre musicale ;*

4) une donnée synthétique visuelle est une modélisation par ordinateur d'une image 2D ou 3D.

La fusion de ces différents objets repose sur une modélisation objet.

EXEMPLE 7.1.– *La vidéo d'un débat entre deux personnes peut être décrite par trois objets vidéo. Deux objets représentent les deux personnages. Le troisième objet représente l'arrière-plan de la scène. A ces trois objets, il est alors possible d'ajouter de nouveaux objets naturels ou/et synthétiques : incrustation d'informations de publicités, etc.*

Ce chapitre se concentre sur le codage vidéo et son extension actuelle (en 2008) MPEG4-AVC. La section 7.1 cite, sans les décrire, les outils et les profils liés aux données synthétiques.

La sous-partie dédiée au codage de vidéos naturelles est divisée en cinq catégories qui fournissent les protocoles des flux binaires et les règles de construction des décodeurs[2].

Avant d'étudier ces cinq catégories, la section 7.2 présente les concepts novateurs de MPEG4. En particulier, elle définit :

1) les *objets vidéo* ;

2) les *plans d'objets vidéo* (VOP) de *formes rectangulaires* ;

3) les plans d'objets vidéo de *formes quelconques* ;

4) la structure définissant la composition d'une scène (BIFS).

Elle présente de manière générale les profils et les niveaux imposés par le standard.

2. Comme pour tous les standards MPEG, MPEG4 décrit parfaitement les flux binaires entre le codeur et le décodeur ainsi que les règles de construction du décodeur, mais, n'impose que le strict nécessaire pour la construction du codeur.

Ensuite, la section 7.3 décrit les profils de codage ne supportant que les VOP de forme rectangulaire. Les profils autorisant le codage des VOP de formes quelconques sont regroupés dans la section 7.4. La notion de granularité, telle qu'elle a été définie par MPEG2 et étendue par MPEG4, est détaillée en section 7.5. La section 7.6 présente les outils dédiés aux codages des images fixes qui peuvent, par exemple, servir au placage de texture pour les images synthétiques 2D ou 3D. Enfin, la section 7.7 explique de manière concise les spécificités des profils dédiés à la production.

Avant la synthèse, la section 7.8 présente le standard actuellement (en 2008) fonctionnel : MPEG4 *Advanced Video Coding* (MPEG4-AVC).

7.1. Codage des données synthétiques

La partie intitulée *extension de modèles d'animations* (traduction libre pour *Animated Framework eXtension* – AFX) gère l'animation des objets synthétiques.

Elle offre plusieurs profils :

1) le profil *Simple Face Animation* (SFA) décrit un modèle de visage pour des applications telles que des audiovisuels de présentation. Un visage générique est définit par le standard. Il comporte des points de contrôle pour les animations. En complément, d'autres points de contrôle servent à préciser l'expression du visage afin de le personnaliser ;

2) le profil *Simple Face and Body Animation* (SFBA) complète le profil SFA en y incluant l'animation d'un corps. Un corps générique est définit par le standard. Comme pour le visage générique, celui-ci comporte des points de contrôle pour l'animation et des points de contrôle pour améliorer la description du corps ;

3) le profil *Basic Animated Texture* (BAT) permet de combiner des modèles synthétiques de visages avec des images fixes codées à l'aide de l'outil VTC (voir section 7.6) ;

4) le profil *Hybrid* (H) reconnaît des objets aussi bien synthétiques que naturels dont il permet la combinaison pour des applications fortement orientées multimédia. Les outils permettant la combinaison d'objets naturels et d'objets synthétiques sont regroupés sous la terminologie *Synthetic and Natural Hybrid Coding* (SNHC) dont une traduction possible est : *codage mixte naturel et synthétique*.

EXEMPLE 7.2.– *On peut coder l'animation d'un visage synthétique et synchroniser l'expression du visage et le mouvement des lèvres avec un codage synthétique de type TTS. La composition de ces deux objets synthétiques peut ensuite s'insérer sur un site de téléachat ou de visite de musée.*

Les objets synthétiques visuels sont des images de synthèse 2D ou 3D modélisées par *triangulation* de leurs surfaces.

La triangulation modélise une image par un ensemble de triangles jointifs qui forment un treillis recouvrant la surface. Les sommets des triangles sont des points spécifiques de la surface, appelés *points de contrôles*. Le déplacement des points de contrôle permet de déformer la surface.

Le placage de texture utilise également ces points de contrôle comme repère. Une texture est un *morceau* d'image naturelle qui vient envelopper la surface décrite par le maillage. Les transformations utilisent les points de contrôle pour amèner la texture à s'adapter à la surface décrite par le treillis.

On peut également – et ceci représente un défi actuel (en 2008) en recherche – modéliser un objet d'une séquence vidéo naturelle par un treillis spatio-temporel (3D) déformable. Ce treillis est en fait un ensemble de treillis 2D. Chaque treillis 2D modélise l'objet dans une image de la séquence vidéo. Deux treillis de deux images successives ont leurs sommets en relation. Cette relation décrit la déformation temporelle de l'objet dans la séquence.

7.2. Généralités sur les objets vidéo

En plus de la synchronisation et du multiplexage déjà présents dans les précédents standards (MPEG1/2), MPEG4 définit un codage *orienté objet* des audiovisuels.

Un audiovisuel devient une collection d'objets agencée à l'aide d'un langage permettant une description arborescente de la scène. Ce langage, appelé *BInary Format for Scene* (BIFS), étend le principe de description de scènes offert par le langage VRML (*Virtual Reality Modeling Language*). En particulier, il prend en compte le temps et la synchronisation pour restituer la dynamique temporelle de l'audiovisuel ainsi codée.

Les objets MPEG4 sont les *objets vidéo* (VO) et les *objets audio* de l'audiovisuel à coder. Ils peuvent être *naturels* ou *synthétiques* :

1) une séquence vidéo (VS), obtenue à l'aide d'une caméra, peut être découpée en plusieurs objets vidéo naturels, décrivant les personnes, les animaux et les objets ayant une importance dans la scène filmée ;

2) les objets vidéo synthétiques sont le résultat d'une modélisation 2D ou 3D faite par ordinateur.

Comme l'illustre la figure 7.1a, un objet vidéo peut être *rectangulaire*. L'objet est alors simplement une séquence d'images ou de sous-images de la vidéo. Chaque (sous-)image de la séquence est un *plan de l'objet vidéo* (VOP). L'objet vidéo peut également être de *forme quelconque* (voir figure 7.1b). L'objet correspond alors à une séquence de régions extraites des images de la vidéo. Mais, MPEG4 ne précise

pas comment sont extraites les régions (manuellement, automatiquement ou par un processus supervisé). La décision est laissée au constructeur du codeur. Il est supposé que les VOP (de forme quelconque) d'un VO décrivent une personne, un animal ou encore un objet (un moulin, par exemple) facilement identifiable.

REMARQUE 7.1.– *Il faut noter que l'extraction de régions représentant des caractéristiques spécifiques (par exemple, un objet vidéo peut décrire une voiture se déplaçant dans la séquence vidéo) n'est pas une tâche aisée.*

Ainsi une VS est décrite hiérarchiquement (voir figure 7.2). Elle est composée de VO. Chaque VO est, à son tour, composé d'un ensemble de *résolutions d'objet vidéo* (VOL).

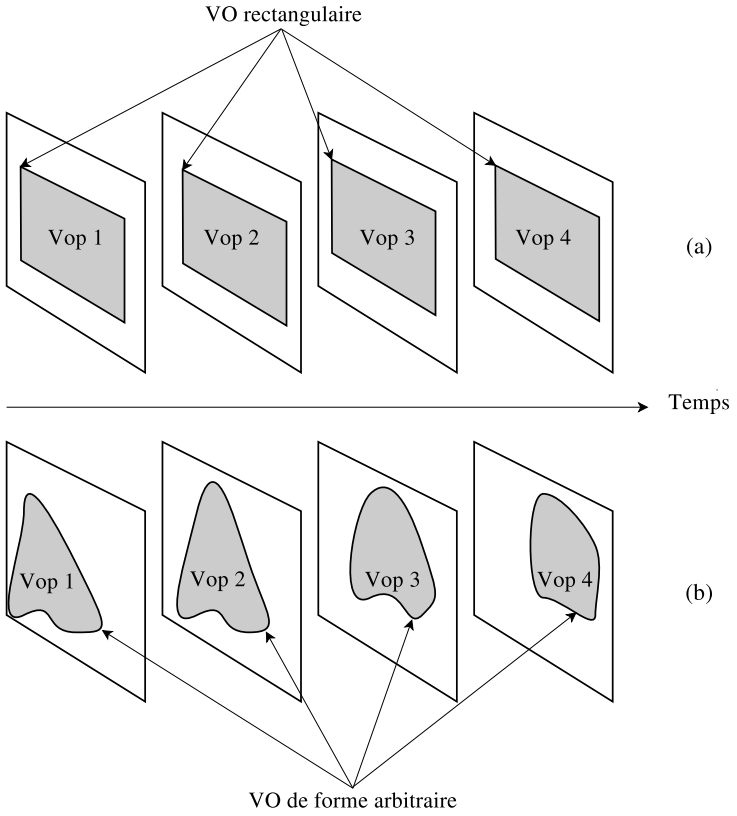


Figure 7.1. *Objets vidéos : (a) VO rectangulaire où chaque VOP est une partie rectangulaire d'une image de la séquence vidéo, (b) VO de forme quelconque où chaque VOP est une région extraire d'une image de la séquence vidéo*

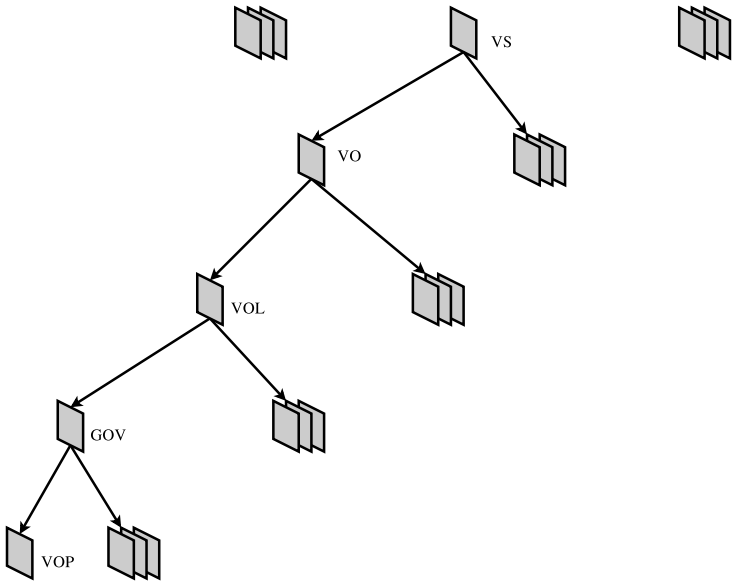


Figure 7.2. *La structure hiérarche d'une vidéo au format MPEG4*

Le format MPEG4 reprenant les notions de *granularité* (spatiale et temporelle) introduites avec le format MPEG2, un VOL décrit un VO à un certain niveau de granularité spatiale et/ou temporelle. Chaque VOL est composé de *groupes de VOP* (GOV). Les GOV s'apparentent aux GOP des formats MPEG1/2 (voir section 5.2). De la même manière, la bande sonore d'un film peut être découpée en *objets naturels audios* : le dialogue de chaque personnage, le son provenant de l'arrière plan de la scène, etc.

A cette collection d'objets naturels peuvent s'ajouter des objets synthétiques. Par exemple, un personnage virtuel peut être ajouté aux objets vidéo naturels. Son déplacement dans la scène est codé à l'aide du BIFS. La figure 7.3 décrit schématiquement l'ensemble de ces objets, leurs codages, la description arborescente de la scène (BIFS) et leur composition après décodage. L'objet *moulin* est ajouté par le client grâce au codage en objets. On voit avec cet exemple que l'interactivité peut intervenir à tous les niveaux : du codeur (côté serveur) au décodeur (côté client).

De plus, le standard propose un environnement de programmation Java : MPEG-J. Cet environnement permet aux applications Java (les *MPEGlets*) d'accéder aux librairies et API Java pour faciliter les interactions avec le codeur et/ou le décodeur. C'est une spécificité du standard MPEG4. En effet, MPEG2 propose une interaction uniquement sur le flux binaire codé (voir DSM-CC en section 6.4). Avec le standard MPEG4,

l'interaction peut intervenir directement sur le codeur. La figure 7.4 compare le niveau d'interactivité entre les formats MPEG2 et MPEG4.

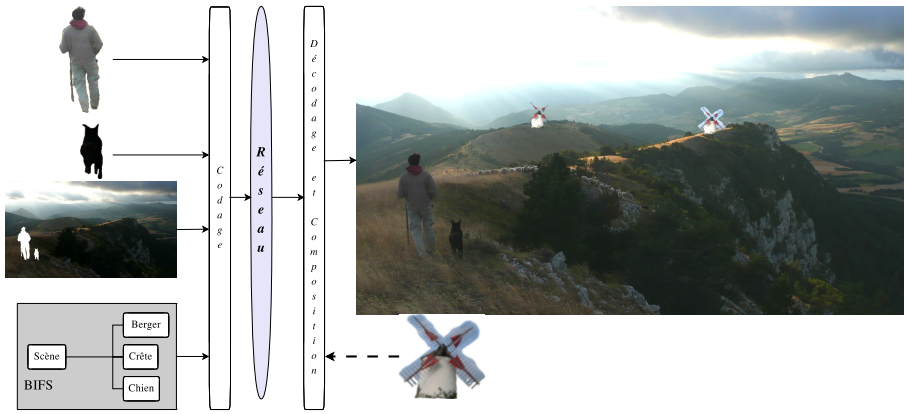


Figure 7.3. *Simulation du fonctionnement général du standard MPEG4*

7.2.1. *Profils et niveaux*

La notion de profil, déjà utilisé par MPEG2, est largement étendue. Quinze profils sont définis et regroupés en cinq catégories.

7.2.1.1. *Catégorie 1*

La première catégorie, décrite en section 7.3, ne prend en compte que les VO rectangulaires :

1) le profil *Simple* (S) propose un codage de faible complexité pour des VO rectangulaires uniquement. Il est étiqueté VLBR (étiquetage expliqué plus loin) car il est dédié au codage pour des débits binaires très faible ;

2) le profil *Advanced Simple* (AS) reprend le profil *Simple* en augmentant son efficacité. De plus, il prend en compte les vidéos entrelacées ;

3) le profil *Advanced Real-Time Simple* (ARTS) correspond au codage *temps-réel* des VO rectangulaires.

7.2.1.2. *Catégorie 2*

La deuxième catégorie, décrite en section 7.4, supporte aussi bien les VO rectangulaires que les VO de forme quelconque :

1) le profil *Core* (C) est le noyau des codeurs pour les VO de forme quelconque ;

2) le profil *Main* (M) ajoute des outils au profil *Core* ;

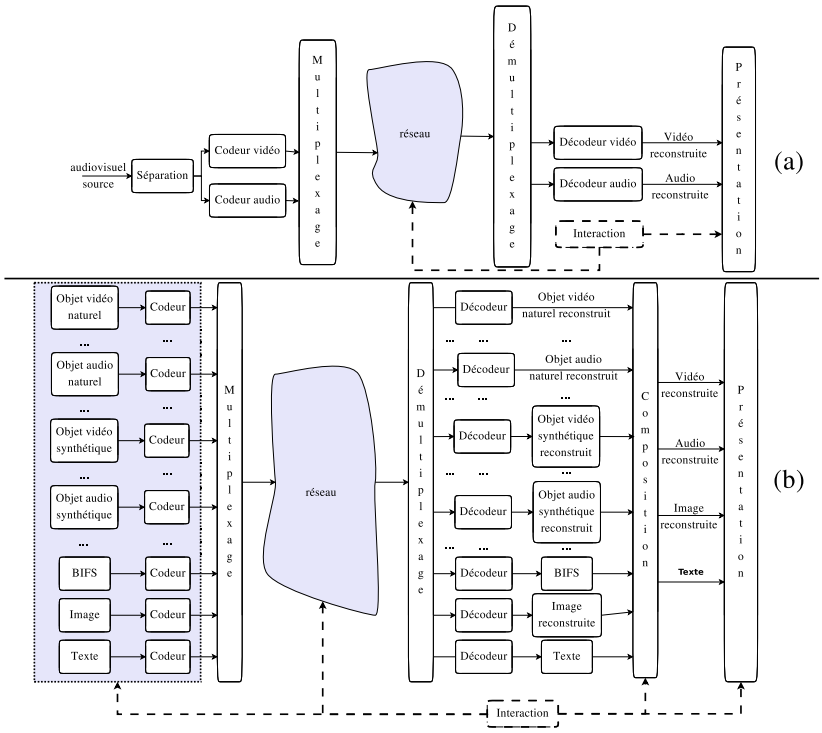


Figure 7.4. *(a) Le modèle MPEG2 où l'interaction intervient au niveau réseau (voir DSM-CC en section 6.4) ; (b) le modèle MPEG4 où l'intercation agit sur le flux binaire, la composition de la scène, le codage et la repésentation de la source, le décodage*

3) le profil *Advanced Coding Efficiency* (ACE) est la version efficace en termes de débit binaire du profil *Core* ;

4) le profil *N-bit* (Nb) est utilisé pour un codage des VO de forme quelconque avec une profondeur autre que 8 bits. C'est également une version efficace du profil *Core* pour les faibles débits ou les débits variables.

7.2.1.3. *Catégorie 3*

La troisième catégorie, décrite en section 7.5, introduit la *granularité temporelle* et la *granularité spatiale* – déjà utilisées dans MPEG2 (voir paragraphe 6.2.5) – ainsi que la *granularité binaire* :

1) le profil *Simple Scalable* (SS) propose la granularité temporelle et la granularité spatiale pour les VO rectangulaires ;

2) le profil *Fine Granular Scalability* (FGS) propose la granularité en plans binaires pour les VO rectangulaires ;

3) le profil *Core Scalable* (CS) propose la granularité spatiale pour les VO de forme rectangulaire et quelconque, mais, propose uniquement la granularité temporelle pour les VO rectangulaires.

7.2.1.4. *Catégorie 4*

La quatrième catégorie, décrite en section 7.6, définit les techniques de codage pour les images qui ne sont pas extraites d'une vidéo :

1) le profil *Scalable Texture* (ST) est dédié aux images fixes. Il peut s'agir de *panoramas* ou bien d'informations annexes à l'audioviduel ;

2) le profil *Advanced Scalable Texture* (AST) augmente l'efficacité du profil *Scalable Texture* et ajoute des outils pour les images fixes de très grandes tailles ;

3) le profil *Advanced Core* (AC) est une combinaison des profils *Simple*, *Core* et *Advanced Scalable Texture*.

7.2.1.5. *Catégorie 5*

La dernière catégorie, section 7.7, regroupe les profils les plus riches en outils. Ils servent principalement à la production et à la distribution entre différents sites de production :

1) le profil *Simple Studio* (SSt) est le profil haute résolution pour les VO de tout type ;

2) le profil *Core Studio* (CSt) augmente le taux de compression du profil *Simple Studio*.

7.2.1.6. *Compabilité entre objets vidéo et profils*

Chaque profil permet de coder (resp. de décoder) « simultanément »[3] un certain nombre d'objets vidéo de différentes catégories à divers *niveaux de résolution* spatiale.

EXEMPLE 7.3.– *Le profil Simple permet de coder (resp. de décoder) jusqu'à quatre objets vidéo Simple avec une résolution* 176×144 *ou* 352×288*. (Le débit binaire varie alors de 64 Kbps à 384 Kbps.) Un objet Simple est un objet vidéo codé à l'aide des outils du profil Simple. Il en va de même pour les autres profils.*

En plus des VO codés à l'aide de leurs outils, certains profils reconnaissent également les VO codés par des profils utilisant un sous-ensemble de leurs outils. Il y a une hiérarchie partielle qui autorise certains profils à reconnaître les objets d'autres profils. Le tableau 7.1 liste les compatibilités existantes.

3. Simultanément signifie que les objets sont présents au même instant dans la mémoire du codeur (ou du décodeur). Par exemple, une image peut être découpée en quatre parties rectangulaires qui seront les VOP rectangulaires de la vidéo. Au même moment, ces quatre VOP sont rangés en mémoire et codés.

EXEMPLE 7.4.– *Le profil Advanced Simple propose le codage (resp. le décodage) de un à quatre objets vidéo Simple ou Advenced Simple. (Le débit binaire varie alors de 128 Kbps à 8 Mbps.)*

Le choix d'un profil, d'une résolution et du nombre d'objets codés permet de s'adapter aux capacités du ou des réseaux connectant le codeur au décodeur. La capacité de mémorisation du décodeur est également une contrainte sur le niveau de résolution possible, le nombre d'objets simultanément décodables et le débit binaire admissible.

Profil Type d'objet	S	AS	ARTS	C	M	ACE	Nb	SS	FGS	CS	ST	AST	AC	SSt	CSt
S	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok			ok		
AS		ok							ok						
ARTS			ok												
C				ok	ok	ok	ok			ok			ok		
M					ok										
ACE						ok									
Nb							ok								
SS								ok		ok					
FGS									ok						
CS										ok					
ST					ok						ok				
AST					ok							ok	ok		
SSt														ok	ok
CSt															ok

Tableau 7.1. *Les objets vidéo reconnus par les profils*

7.3. Le codage des formes rectangulaires

Cette catégorie est dédiée au codage des VO rectangulaires. Le format vidéo en entré est souple : échantillonnage 4:4:4:, 4:2:2 ou 4:2:0 en mode progressif ou entrelacé (voir sections 5.2 et 6.2). Les VOP sont les images ou les trames de la vidéo source. Chaque VOP est codé différemment suivant qu'il est un I-VOP, un P-VOP ou un B-VOP.

7.3.1. *I-VOP*

Un I-VOP rectangulaire est prédit en *mode intra* (voir section 7.3.1.1) : aucun autre VOP que celui en cours de codage n'intervient dans la prédiction. La figure 7.5 schématise les étapes du codeur :

1) la DCT opère sur des blocs de taille 8×8 (voir section 3.1) ;

2) le module Q quantifie les coefficients du bloc fréquentiel provenant de la DCT. Il est gouverné par le pas de quantification M, qui varie entre 1 et 31 :

a) la quantification q_{DC}, d'un coefficient DC, est régie par la règle suivante :

$$q_{\mathrm{DC}} = \mathrm{round}\left(\frac{\mathrm{DC}}{s_{\mathrm{DC}}}\right)$$

où s_{DC} est fixé à l'aide du paramètre de quantification M et du tableau 7.2,

b) les coefficients AC sont quantifiés suivant la méthode utilisée par le standard MPEG1 (voir equation (5.1) page 114). Pour le standard MPEG4, la matrice de pondération S, par défaut, est donnée par le tableau 7.3 ;

3) la mise en forme ordonne les coefficients par un parcours en zigzag puis effectue un codage par plages des valeurs nulles (voir section 3.1). Le format MPEG4 propose trois parcours en zigzag, décrits en figure 7.6 ;

4) le codage entropique est un codage à longueur variable similaire au codeur de Huffman.

DÉFINITION 7.2.– *Le codage qui vient d'être décrit, est appelé* codage en texture *car il est utilisé aussi bien pour le codage en mode* intra *ou mode* inter*, que pour le codage d'objets spécifiques (définis par des profils spécifiques).*

La texture *représente soit les intensités, soit les différences d'intensités suivant le cas de figure.*

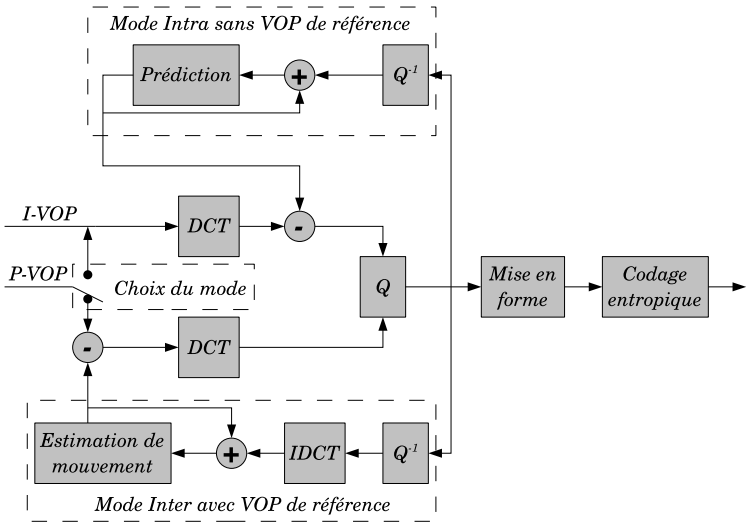


Figure 7.5. *Codage des I-VOP et des P-VOP*

	$M \leqslant 4$	$5 \leqslant M \leqslant 8$	$9 \leqslant M \leqslant 24$	$25 \leqslant M$
s_{DC} pour les blocs Y	8	$2M$	$M+8$	$2M-16$
s_{DC} pour les blocs Cb et Cr	8	$\frac{M+13}{2}$	$\frac{M+13}{2}$	$M-6$

Tableau 7.2. *Les valeurs s_{DC}, en fonction du paramètre M*

8	47	18	19	21	23	25	27
17	18	19	21	23	25	27	28
20	21	22	23	24	26	28	30
21	22	23	24	26	28	30	32
22	23	24	26	28	30	32	35
23	24	26	28	30	32	35	38
25	26	28	30	32	35	38	41
27	28	30	32	35	38	41	45

Tableau 7.3. *Matrice de pondération par défaut du standard MPEG4*

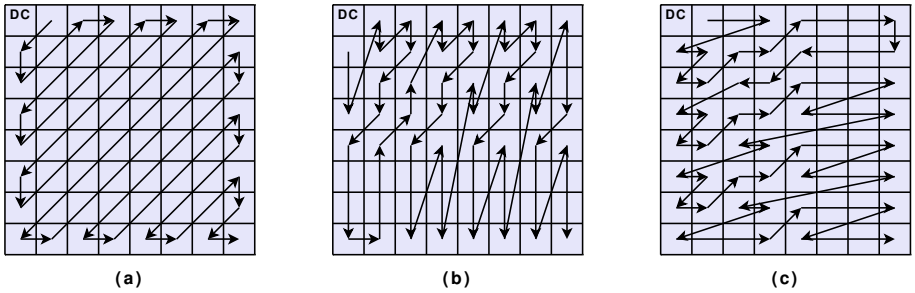


Figure 7.6. *Les trois parcours en zigzag proposés : (a) classique (MPEG1), (b) et (c) sont utilisées pour certains types de VOP tels que les vidéos entrelacées*

7.3.1.1. *Le mode* intra *de la prédiction*

Cette option permet d'intégrer une forme de prédiction des macroblocs pour les I-VOP. Dans les standards MPEG1/2, les I-images (les équivalents des I-VOP) sont codées sans aucune prédiction. Toutefois, la corrélation entre blocs (de taille 8×8) voisins peut être prise en compte. C'est ce que propose le standard MPEG4.

Dans la figure 7.7, le bloc X du macrobloc en cours de codage peut être prédit à partir des blocs voisins, A, B et C. Seuls le coefficients DC (en noir dans la figure) et les coefficients AC de la première ligne et de la première colonne (en gris sur la figure) sont prédits.

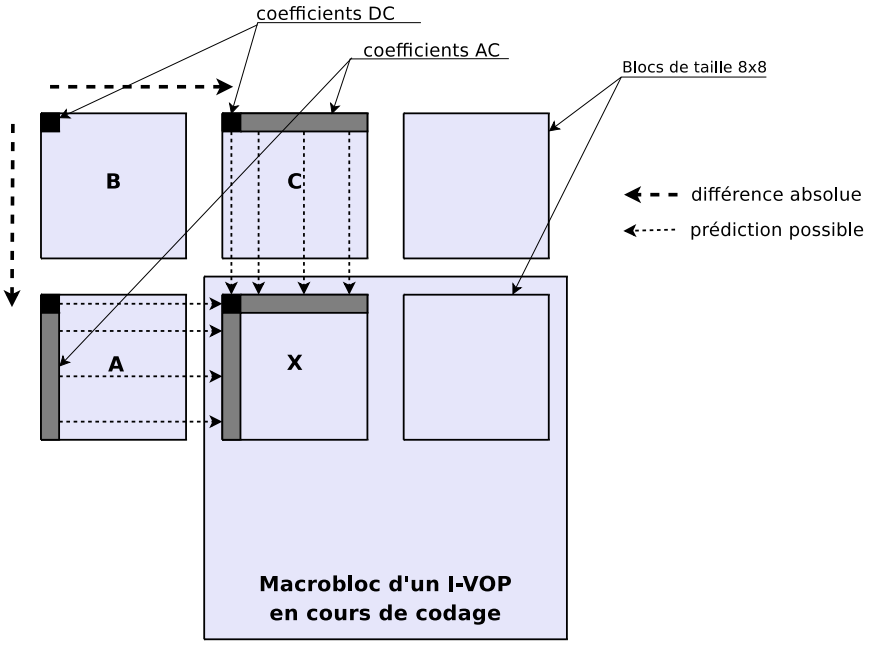


Figure 7.7. *Prédiction* intra *des coefficients DC*

La prédiction $\widetilde{\mathrm{DC}}_X$, du coefficient DC est faite en fonction des différences absolues des coefficients, $\widehat{\mathrm{DC}}_A$, $\widehat{\mathrm{DC}}_B$ et $\widehat{\mathrm{DC}}_C$:

$$\widetilde{\mathrm{DC}}_X = \begin{cases} \widehat{\mathrm{DC}}_C & \text{si } |\widehat{\mathrm{DC}}_C - \widehat{\mathrm{DC}}_B| < |\widehat{\mathrm{DC}}_B - \widehat{\mathrm{DC}}_A| \\ \widehat{\mathrm{DC}}_A & \text{sinon} \end{cases}$$

Les blocs A, B et C ont été codés auparavant, puis, décodés pour la prédiction. Les valeurs décodées sont donc les coefficients quantifiés. L'erreur de prédiction transmise au décodeur est la différence entre la prédiction et la valeur quantifiée de DC_X.

La différence absolue minimale indique la direction de prédiction choisie. Cette direction est conservée pour la prédiction des coefficients AC. Ainsi, lorsque la prédiction $\widehat{\mathrm{DC}}_X$ vaut $\widehat{\mathrm{DC}}_C$ (resp. $\widehat{\mathrm{DC}}_A$), les coefficients (quantifiés) de la première ligne du bloc C (resp. de la première colonne du bloc A) forment les prédictions des coefficients de la première ligne (resp. de la première colonne) du bloc X.

7.3.2. *P-VOP*

Un P-VOP rectangulaire est prédit :

1) soit en mode *intra* comme les I-VOP ;

2) soit en mode *inter* à partir d'un I-VOP ou P-VOP antérieur déjà codé.

Le mode *inter* est plus efficace en termes de débit binaire mais le mode *intra* peut s'avérer avantageux pour le codage de macroblocs dont la prédiction en mode *inter* est médiocre. La figure 7.5 en donne le fonctionnement général.

La compensation de mouvement utilise les I-VOP et les P-VOP déjà codés qu'elle décode. L'estimation est faite avec une précision de l'ordre du demi-pixel ou du quart de pixel suivant le profil. Les valeurs entre les pixels sont calculées par une technique d'interpolation qui est laissée à la discrétion du concepteur (voir annexe D.1). L'erreur d'estimation est ensuite codée suivant le même principe que celui utilisé pour les macroblocs des I-VOP : DCT, quantification, mise en forme et codage entropique.

7.3.3. *B-VOP*

Un B-VOP rectangulaire est prédit en *mode inter* à l'aide de deux VOP de référence, respectant les deux propriétés suivantes :

1) chacun de ces VOP de référence est soit un I-VOP soit un P-VOP ;

2) un des VOP est de référence antérieure tandis que l'autre est de référence postérieure.

Quatre types de prédiction sont possibles :

1) la *prédiction en avant* n'utilise que le VOP de référence antérieure (voir paragraphe 6.2.1) ;

2) la *prédiction en arrière* n'utilise que le VOP de référence postérieure (voir paragraphe 6.2.1) ;

3) la *prédiction bidirectionnelle* utilise les deux VOP de référence (voir paragraphe 6.2.1). L'estimation de mouvement fournit deux vecteurs de déplacement. La moyenne des deux macroblocs de référence forme la prédiction ;

4) la *prédiction directe* utilise également deux VOP de référence. Cependant, le vecteur de la prédiction antérieure MV_F, et le vecteur de la prédiction postérieure MV_B, ne sont pas directement codés. Grâce à la corrélation supposée entre les VOP, ces vecteurs sont linairement définis à partir du vecteur MV, qui est l'estimation du macrobloc du P-VOP postérieur situé à la même position que $\mathcal{B}$ (voir figure 7.8). Le

Le codeur transmet un unique *vecteur différentiel* D_{MV} :

$$\begin{aligned} \mathrm{MV}_F &= \frac{T_{\mathcal{B}A}}{T_{\mathcal{B}A}+T_{P\mathcal{B}}} \cdot \mathrm{MV} + D_{\mathrm{MV}} \\ \mathrm{MV}_B &= \frac{T_{P\mathcal{B}}}{T_{\mathcal{B}A}+T_{P\mathcal{B}}} \cdot \mathrm{MV} + D_{\mathrm{MV}} \end{aligned}$$

avec :

$$\begin{aligned} T_{\mathcal{B}A} &= \text{date\_affichage}(\mathcal{B}) - \text{date\_affichage(VOP ref. antérieure)} \\ T_{P\mathcal{B}} &= \text{date\_affichage(VOP ref. postérieure)} - \text{date\_affichage}(\mathcal{B}) \end{aligned}$$

D_{MV} est le vecteur de recalage transmit au décodeur. Ceci suppose que le mouvement entre les deux références est linéaire.

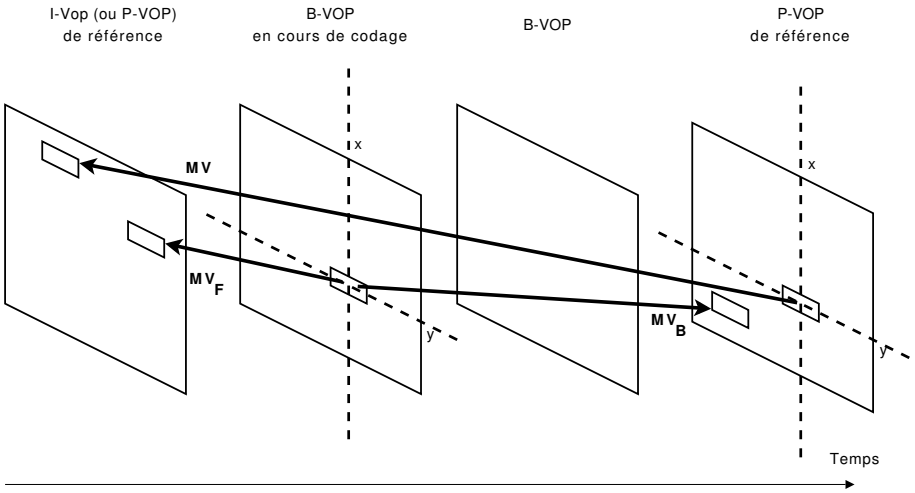


Figure 7.8. *Prédiction en mode direct des B-VOP*

EXEMPLE 7.5.– *Pour l'exemple de la figure 7.8 :*

$$\begin{aligned} \mathrm{MV}_F &= \tfrac{1}{3}\,\mathrm{MV} + D_{\mathrm{MV}} \\ \mathrm{MV}_B &= -\tfrac{2}{3}\,\mathrm{MV} + D_{\mathrm{MV}} \end{aligned}$$

7.3.3.1. *Mode silence*

Si le macrobloc est codé en mode silence[4], la compensation de mouvement utilise la prédiction directe avec un vecteur différentiel (D_{MV}) nul. De plus, aucune erreur de prédiction n'est enregistrée.

Le gain notable en termes de débit se fait au détriment de la qualité puisque le décodeur reconstruit le macrobloc à l'aide de la seule prédiction définie avec un vecteur différentiel nul.

Ce mode est adapté aux zones (quasi) uniformes mais doit être évité quand le macrobloc à coder présente une forte dynamique en intensité ou s'il provient d'une zone soumise à un fort mouvement.

7.3.4. *Le profil* Simple

Le profil S est connu comme un codeur à *débit binaire très faible*, traduction libre pour *Very Low Bit Rate* (VLBR) : la téléphonie mobile, par exemple.

Ce profil regroupe (voir tableau 7.4) :

1) les objets I-VOP, P-VOP et «paquets vidéo» ;

2) les outils 4MV, UMV, «prédiction *intra*», «prédiction au demi-pixel», regroupement et codeur à longueur variable réversible (RVLC).

Les objets I-VOP, P-VOP, la vidéo progressive et les options « prédiction *intra* » et « prédiction au demi-pixel » sont décrits ci-dessus. Les options 4MV et UMV proposent des améliorations en termes de qualité de prédiction des P-VOP. (Et donc en termes de débit binaire.) Les objets « paquets vidéo » et les options regroupement et RVLC sont des aides à la correction d'erreurs de transmission. Le but est de limiter la propagation d'une erreur de décodage ou d'une perte d'information. Si elle n'est pas corrigée, cette erreur se propage aussi bien spatialement (dans le VOP en cours de décodage) que temporellement *via* les prédictions faites lors du codage (une erreur lors du décodage d'un VOP de référence a des répercussions sur d'autres VOP).

7.3.4.1. *L'option 4MV* (4 Motion Vectors)

L'option 4MV signifie que la prédiction d'un macrobloc utilise le macrobloc en entier ou bien qu'elle découpe le macrobloc en quatre blocs de taille 8×8 :

1) si la prédiction est faite sur le macrobloc en entier, un seul vecteur de déplacement est transmis ;

4. Traduction libre pour *skipped mode*.

Profil S
I-VOP
P-VOP
Paquets vidéo
Vidéo progressive
4MV
UMV
Prédiction *intra*
Prédiction au demi-pixel
Partitionnement
RVLC

Tableau 7.4. *Les objets et outils du profil S*

2) si le macrobloc est scindé en quatre blocs, une prédiction est faite pour chacun des blocs. On obtient donc au total quatre vecteurs de déplacement au lieu d'un seul.

La figure 7.9 illustre le découpage et les prédictions qui en résultent. Le choix du mode de prédiction (1MV ou 4MV) repose sur le gain énergétique (un critère MAD par exemple) obtenu en effectuant le découpage et sur le débit binaire supporté.

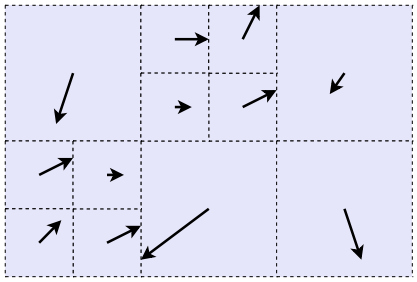


Figure 7.9. *Certains macroblocs sont prédits en mode 1MV, d'autres en mode 4MV*

7.3.4.2. *L'option UMV* (Unrestricted Motion Vectors)

L'option UMV permet de faire des prédictions plus fines lorsque le macrobloc estimé (de taille 16×16) s'étend au-delà des limites du VOP de référence auquel il appartient. Le standard MPEG4 a choisi de dupliquer les valeurs en bordure du VOP (rectangulaire) sur les pixels extérieurs du macrobloc. La figure 7.10 illustre le procédé. On parle de *padding* qui peut littéralement se traduire par *remplissage* ou *bourrage*.

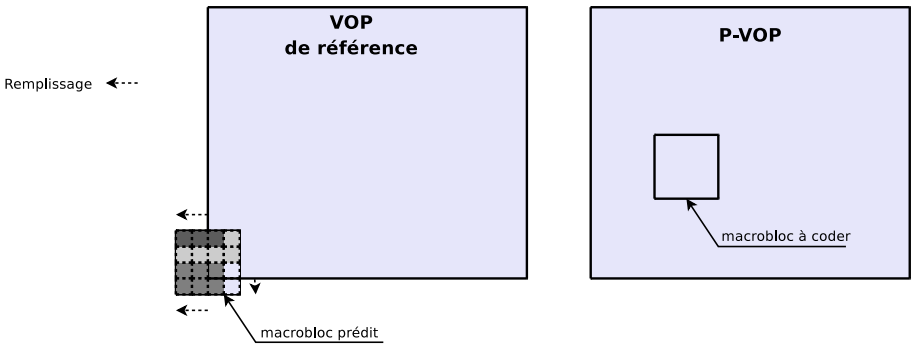


Figure 7.10. *Remplissage des pixels du macroblocs se situant au-delà des limites du VOP*

7.3.4.3. *Les « paquets vidéo »*

Les « paquets vidéo » sont assez similaires aux bandes (*slices*) du format MPEG1 (voir section 5.2). Toutefois, le terme de bande (*slice*) donnant lieu à une définition légèrement différente dans la partie AVC (partie 10) de MPEG4, on préfère parler de «paquets vidéo». Un VOP est découpé en plusieurs « paquets vidéo ». Chaque « paquet vidéo » regroupe plusieurs macroblocs et est muni d'un marqueur de synchronisation. Ce marqueur est suivi d'un compteur pour le numéro du prochain macrobloc. Il permet au décodeur de positionner corrrectement le premier macrobloc des « paquets vidéo », en toutes circonstances.

7.3.4.4. *Le regroupement*

Le regroupement reprend le principe déjà utilisé par le format MPEG1 (voir paragraphe 6.2.5). Chaque « paquet vidéo » est scindé en deux groupes de coefficients[5] :

1) le premier groupe contient le mode de prédiction choisi (*intra* ou *inter*), suivi des coefficients DC (mode *intra*) ou des vecteurs de déplacement (mode *inter*) de l'ensemble des macroblocs du Paquet Vidéo ;

2) le deuxième groupe rassemble tous les coefficients AC des macroblocs du « paquet vidéo » ainsi que les coefficients DC pour le mode *inter*.

7.3.4.5. *Le codeur à longueur variable réversible (RVLC)*

Pour minimiser l'impact des erreurs survenant au niveau des bits du flux binaire, il est proposé d'utiliser un codeur à longueur variable qui permet de décoder une valeur en la lisant aussi bien du bit de poids fort vers le bit de poids faible que du bit de poids faible vers le bit poids fort.

5. On parle ici de coeffcients (provenant de la DCT), mais, dans les faits, les informations codées sont les erreurs de prédiction (en mode *intra* ou en mode *inter*) faites sur ces coefficients.

7.3.5. *Le profil* Advanced Simple

Le profil AS hérite du profil S auquel il ajoute (voir tableau 7.5) :

1) les objets B-VOP et la vidéo entrelacée ;

2) les outils « prédiction au quart de pixel », quantification alternative et compensation globale de mouvement (GMC).

Profil AS :: Profil S
B-VOP
Vidéo entrelacée
Prédiction au quart de pixel
Quantification alternative
GMC

Tableau 7.5. *Les objets et outils du profil AS :*
Profil AS :: Profil S *signifie que le profil AS hérite des objets et des outils du profil S*

7.3.5.1. *La quantification alternative*

Le profil AS propose une deuxième méthode de quantification. Celle-ci n'utilise pas de matrice de pondération, contrairement à la première méthode (issue du format MPEG1). Son algorithme est plus aisé mais d'efficacité moindre. Tous les coefficients, DC et AC, suivent la même règle :

$$q_{\mathrm{AC}} = \begin{cases} \mathrm{sign}(\mathrm{AC}) \left\lfloor \frac{|\mathrm{AC}|}{2M} \right\rfloor & \text{pour les blocs en mode } \textit{intra} \\ \mathrm{sign}(\mathrm{AC}) \left\lfloor \frac{|\mathrm{AC}|-M/2}{2M} \right\rfloor & \text{pour les blocs en mode } \textit{inter} \end{cases}$$

où M est le pas de quantification déjà utilisé par la première méthode. La quantification inverse s'écrit :

$$\widehat{\mathrm{AC}} = \begin{cases} 0 & \text{si } q_{\mathrm{AC}} = 0 \\ (2q_{\mathrm{AC}} + 1) \cdot M & \text{si } M \text{ est impair} \\ (2q_{\mathrm{AC}} + 1) \cdot M - 1 & \text{si } M \text{ est pair} \end{cases}$$

7.3.5.2. *La compensation globalde mouvement (GMC)*

Pour la prédiction d'un P-VOP ou d'un B-VOP, le mouvement de la caméra peut être pris en compte pour l'ensemble de tous les macroblocs du VOP. Par exemple, un zoom arrière de la caméra implique une transformation (homothétie) identique pour l'ensemble des pixels du VOP. En plus de ce mouvement global, la scène peut être localement en mouvement. Un personnage ou un objet peut se déplacer. Aussi, au

niveau de chaque macrobloc, ce mouvement global est mis en compétition avec la compensation de mouvement local classique.

La compensation de mouvement global (GMC) proposée par MPEG4 est un ensemble de un à quatre vecteurs de déplacement global (GMV) muni de leurs positions (voir figure 7.11). Cet ensemble est enregistré dans le flux binaire pour que le décodeur puisse l'utiliser également.

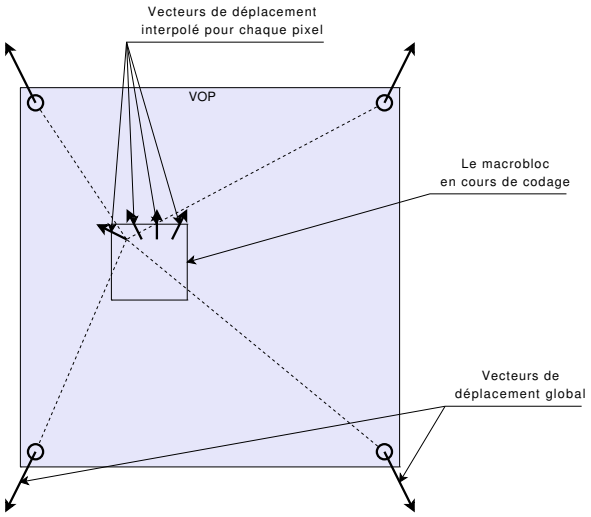


Figure 7.11. *La compensation de mouvement global d'un macrobloc*

La compétition entre la compensation de mouvement local et la compensation de mouvement global, signifie que les GMV doivent être interpolés au niveau de chaque pixel du macrobloc en cours de codage. La technique d'interpolation est indiquée dans le flux binaire pour que le décodeur puisse l'utiliser.

L'erreur de prédiction due aux GMV est comparée à celle de la compensation locale. Le minimum indique quelle technique de compensation est utilisée pour le macrobloc en cours de codage.

7.3.6. *Le profil* Advanced Real-Time Simple

Le profil ARTS hérite du profil S auquel il ajoute les outils conversion dynamique de résolution (DRC) et assigner une nouvelle prédiction (NEWPRED) (voir tableau 7.6).

Profil ARTS :: Profil S
DRC
NEWPRED

Tableau 7.6. *Les objets et outils du profil ARTS*

7.3.6.1. *La conversion dynamique de résolution*

Lorsque le débit binaire augmente de manière significative, il faut prévenir tout risque de dépassement de la limite fixée par le réseaux ou le décodeur.

Par exemple, lors de brusques mouvements dans la scène, la prédiction diminue en qualité (l'erreur de prédiction augmente).

En réduisant de moitié la résolution spatiale pour le codage du (des) VOP susceptible(s) de provoquer un dépassement de capacité, la taille des macroblocs est doublée. L'erreur de prédiction d'un macrobloc couvre alors une zone double dans le VOP original. Cette donnée résiduelle est sous-échantillonnée de moitié avant d'être envoyée au décodeur.

Le décodeur reçoit le VOP à *résolution réduite* (RR). Chaque macrobloc (de taille 16×16) est décodé et suréchantillonné de façon à couvrir la zone du VOP à la résolution originelle (taille 32×32). Ensuite, le *super*-macrobloc (de taille 32×32) du VOP de référence est ajouté.

Les vecteurs de déplacement sont enregistrés à la résolution réduite. Le décodeur doit donc doubler leurs tailles pour retrouver la position exacte du *super*-macrobloc.

La figure 7.12 illustre le fonctionnement du décodeur.

7.3.6.2. *Assigner une nouvelle prédiction*

Cette option autorise le codeur à choisir un VOP de référence de manière à limiter la propagation temporelle d'une erreur qui n'aurait pas été correctement corrigée.

Lorsque le décodeur ne peut pas corriger une erreur, il envoie un *message d'alerte* au codeur signifiant le paquet vidéo à l'origine de cette erreur. Le codeur sélectionne alors un VOP de référence se situant avant ce paquet. Puis, il effectue la compensation de mouvement avec cette nouvelle référence. La propagation temporelle de l'erreur est alors arrêtée. La figure 7.13 illustre le procédé.

7.4. Le codage des formes quelconques

Avec les profils *Core* (C), *Main* (M), *Advanced Coding Efficiency* (ACE) et *N-Bit* (Nb), les VOP sont de forme quelconque. Un tel VOP est montré en figure 7.14a. A

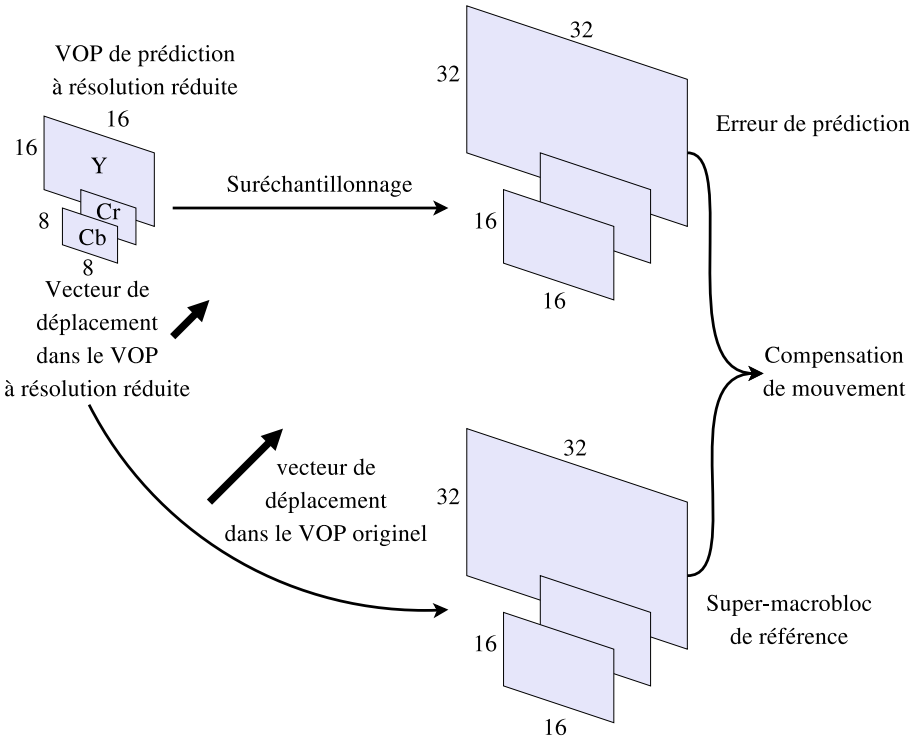


Figure 7.12. *Décodage d'un macrobloc provenant d'un VOP à résolution réduite*

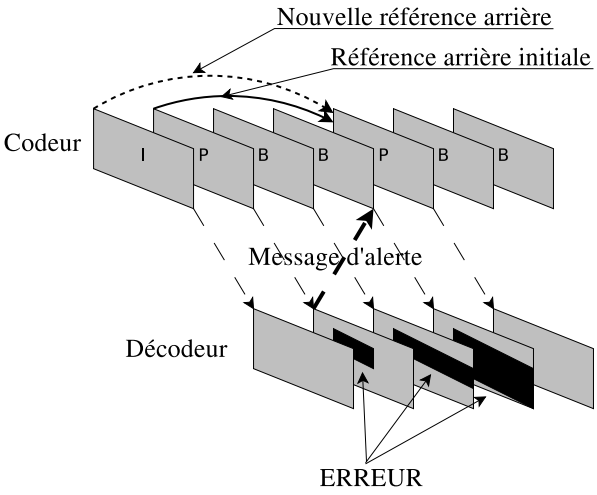


Figure 7.13. *Limitation de la propagation temporelle d'erreurs avec l'outil NEWPRED*

chaque VOP est associé un *rectangle englobant* qui fournit les limites de son support. Afin de faciliter le codage (et le décodage), la taille de ce rectangle englobant est proportionnelle à la taille des macroblocs.

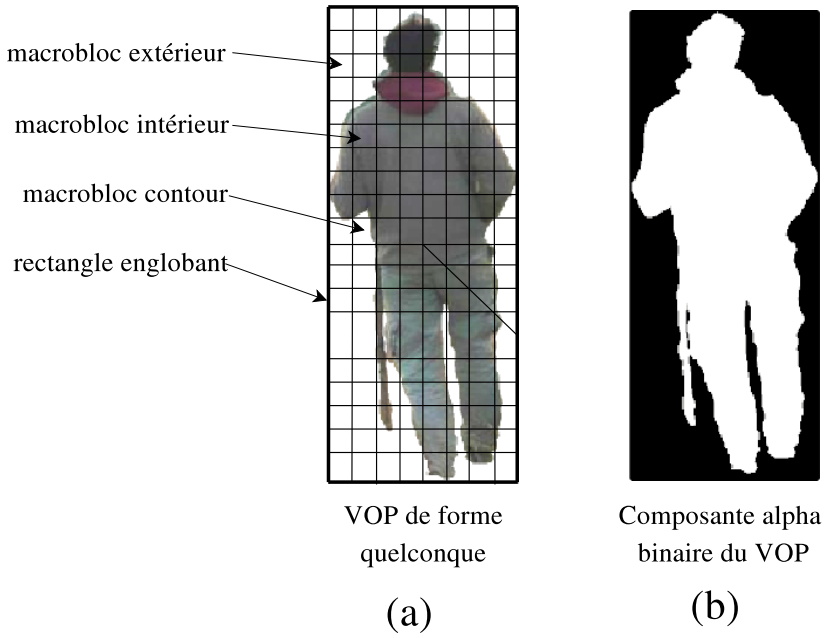


Figure 7.14. *(a) Un VOP de forme quelconque et (b) sa composante* alpha *binaire*

La *forme* (ou contour) d'un VOP est enregistrée comme une *composante alpha binaire* de même taille que le rectangle englobant. Comme pour le format GIF (voir section 2.1), elle prend :

– la valeur 0 quand le pixel est à l'extérieur du VOP (le pixel est totalement transparent) ;

– la valeur 1 quand le pixel est à l'intérieur du VOP (le pixel est totalement opaque).

La figure 7.14b montre la composante *alpha* binaire du VOP en figure 7.14a. Si un macrobloc ne contient que des pixels à l'extérieur du VOP, il est appelé *macrobloc extérieur*. Un *macrobloc intérieur* ne contient que des pixels à l'intérieur du VOP. Tous les autres macroblocs sont des *macroblocs contours*. Ils contiennent des pixels extérieurs et intérieurs.

7.4.1. *Le codage des masques* alpha *binaires (BAB)*

Pour chaque macrobloc contour, il faut enregistrer un *masque alpha binaire* (BAB) permetttant de retrouver le contour du VOP au sein du macrobloc. Ce masque est codé à l'aide d'un codeur artihmétique binaire (voir annexe B du volume 1) puisque ses pixels valent 0 ou 1.

Le codage de chaque pixel est fait en fonction du contexte, c'est-à-dire de ses pixels voisins. Le voisinage est spatial pour le codage *intra* (voir figure 7.15a) et spatiotemporel pour le codage *inter* (voir figure 7.15b). Les valeurs de voisinage sont recueillies dans l'ordre indiqué et forment, ensemble, un mot (de 10 bits pour le codage *intra* et de 9 bits pour le codage *inter*). Ce mot est une entrée dans la table des contextes qui délivre les valeurs des MPS pour la valeur 0.

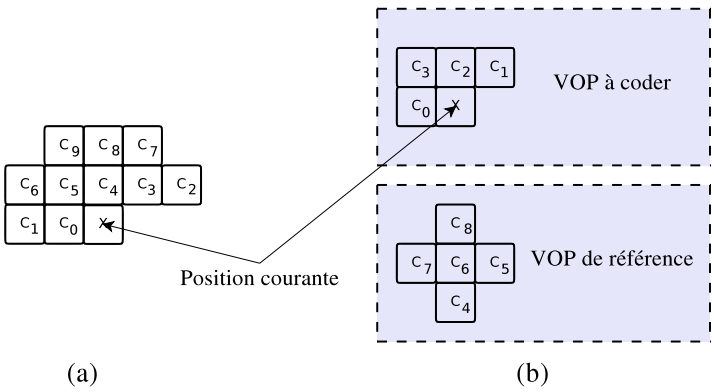


Figure 7.15. *Définition du voisinage (a) spatial pour le codage intra, (b) spatiotemporel pour le codage* inter *où le pixel, du VOP de référence, indicé* c_6 *est à la même position que le pixel en cours de codage (marqué x)*

7.4.2. *Le codage des macroblocs extérieurs et intérieurs*

Les macroblocs extérieurs ne sont pas codés et les macroblocs intérieurs sont codés suivant le principe du codage en texture (voir définition 7.2 page 167) : prédiction, transformée en cosinus discrète, quantification, parcours en zigzag puis codage entropique. La prédiction est en mode *intra* si le macrobloc appartient à un I-VOP. Elle est en mode *inter* s'il s'agit d'un macrobloc d'un P-VOP ou d'un B-VOP.

7.4.3. *Le codage des macroblocs contours*

Les macroblocs contours requièrent un codage spécialisé puisqu'ils contiennent des pixels à l'intérieur et à l'extérieur du VOP. Un prétraitement vient *remplir* les

pixels extérieurs du macrobloc contour. Puis, les techniques des macroblocs intérieurs sont utilisées.

Le remplissage (*padding*) opère en deux passes : une horizontale et une verticale. Lors de la première passe, chaque ligne est traitée indépendamment des autres lignes. Pour une ligne donnée, chaque séquence de pixels extérieurs est détectée comme étant (voir figure 7.16) :

1) soit encadrée par deux pixels intérieures. Alors, les pixels extérieurs de la séquence prennent pour valeur la moyenne des valeurs des deux pixels intérieurs ;

2) soit encadrée par un pixel intérieur d'une part et par un bord du macrobloc d'autre part. Alors, la valeur du pixel intérieur est recopiée sur chacun des pixels extérieurs de la séquence ;

3) soit encadrée par les bords du macrobloc. Dans ce cas, aucun traitement n'est effectué.

La deuxième passe effectue exactement le même traitement mais sur les colonnes du macrobloc après le remplissage des lignes.

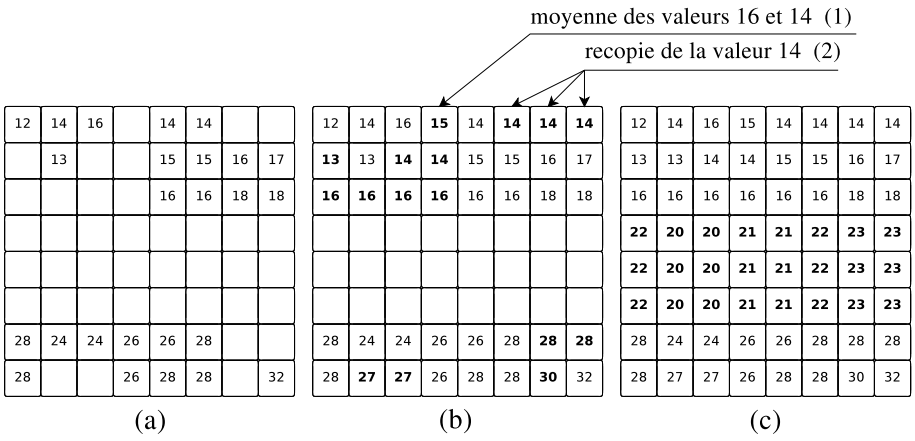


Figure 7.16. *Technique de remplissage (*padding*) : (a) le macrobloc de référence original ; (b) remplissage des lignes ; (c) remplissage des colonnes. Les valeurs résultant du remplissage sont en gras*

7.4.4. *La prédiction de mouvement*

En mode de prédiction *inter*, les macroblocs intérieurs utilisent l'estimation de mouvement déjà définie par le profil *Simple* (voir paragraphe 7.3.4). En revanche,

les macroblocs contours sont traités différemment afin que les prédictions soient correctes. Le codage de l'erreur de prédiction d'un macrobloc contour est identique au codage de l'erreur de prédiction d'un macrobloc intérieur. Mais, l'estimation de mouvement se fait uniquement avec les pixels intérieurs du macrobloc en cours de codage.

Si la prédiction d'un macrobloc, intérieur ou contour, se fait avec un macrobloc contour ou un macrobloc extérieur du VOP de référence, il faut alors fixer des valeurs pour les pixels extérieurs de ce macrobloc de référence.

7.4.4.1. *Le remplissage d'un macrobloc contour de référence*

Si le macrobloc de référence est un macrobloc contour, il ne faut pas que ses pixels extérieurs n'interfèrent dans la prédiction. Pour limiter leur influence, les pixels extérieurs sont valués suivant la technique de remplissage (*padding*) précédemment décrite.

7.4.4.2. *Le remplissage d'un macrobloc extérieur de référence*

Une fois les macroblocs contours du VOP de référence traités, les macroblocs extérieurs sont également remplis. Le principe est récurrent : le remplissage se fait par propagation.

Un macrobloc extérieur juxtaposant un seul macrobloc contour est rempli avec les valeurs du macrobloc contour. Si le bord commun entre les deux macroblocs est horizontal (resp. vertical), la ligne (resp. la colonne) du macrobloc contour est recopiée sur chacune des lignes (resp. des colonnes) du macrobloc extérieur (voir figure 7.17a).

Pour les macroblocs extérieurs ayant plusieurs macroblocs contours comme voisins, le choix du macrobloc contour à recopier se fait suivant la priorité indiquée par la figure 7.17b.

Ensuite, les macroblocs contours et les macroblocs extérieurs remplis servent à remplir les macroblocs restant.

7.4.5. *Le profil* Core

Le profil C hérite du profil S (voir paragraphe 7.3.4) auquel il ajoute (voir tableau 7.7) :

1) les B-VOP ;

2) les composantes *alpha* binaires ;

3) la quantification alternative (voir paragraphe 7.3.5) ;

4) la granularité temporelle en P-VOP.

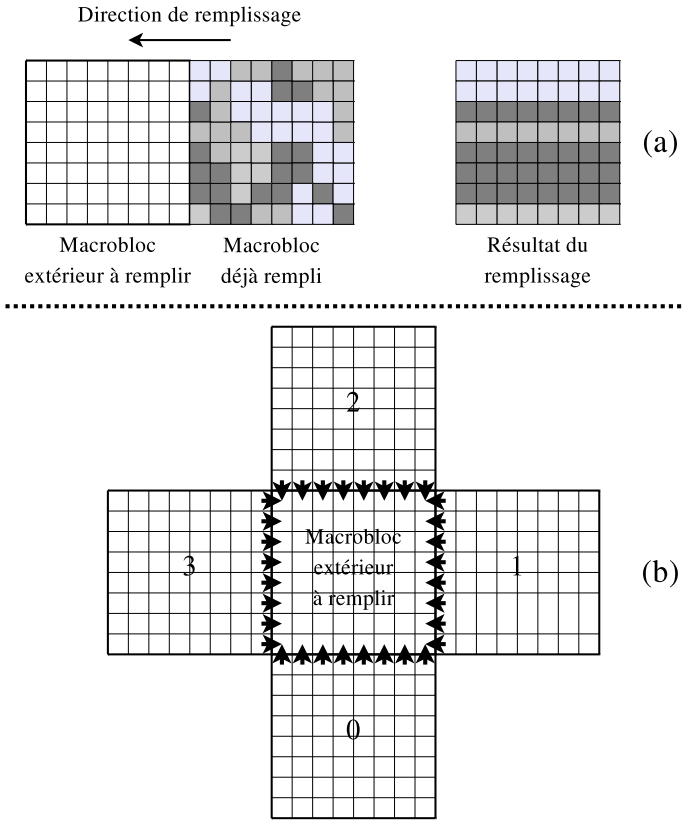


Figure 7.17. *Technique de remplissage des macroblocs extérieurs ayant (a) un seul voisin, (b) plusieurs voisins*

Profil C :: Profil S
B-VOP
Composantes *alpha* binaires
Quantification alternative
Granularité temporelle en P-VOP

Tableau 7.7. *Les objets et outils du profil C*

7.4.5.1. *La granularité temporelle en P-VOP*

La granularité temporelle est identique à celle du standard MPEG2 (voir paragraphe 6.2.5). A partir de la source vidéo, deux flux binaires sont générés. Le flux principal correspond au codage de la source avec un taux de rafraîchissement plus faible que celui de la vidéo source. L'ensemble des outils et des objets du profil C sont utilisés. Le flux complémentaire apporte les images permettant la reconstruction à l'identique (en termes de taux de rafraîchissement) de la source vidéo. Ce flux ne contient que des P-VOP. Les I-VOP du flux principal et les P-VOP des deux flux servent de références aux prédictions du flux complémentaire.

7.4.6. *Le profil* Main

Le profil M hérite du profil C auquel il ajoute (voir tableau 7.8) :

1) les composantes *alpha* ;

2) la vidéo entrelacée ;

3) les panoramas.

Profil M :: Profil C
Composantes *alpha*
Vidéo entrelacée
Panoramas

Tableau 7.8. *Les objets et outils du profil M*

7.4.6.1. *Les composantes* alpha

Les *composantes alpha* sont une extension des composantes alpha binaires.

Les valeurs permises ne sont plus uniquement *totalement transparent* (valeur 0) et *totalement opaque* (valeur 1) mais toute une palette de transparence gérée par des valeurs comprises entre 255 (pour complètement opaque) et 0 (pour complètement transparent).

Une composante *alpha* est codée à l'aide de deux entités :

1) un *masque alpha binaire* pour enregistrer le contour du VOP. Comme pour une composante *alpha* binaire, ce masque prend la valeur 0 pour indiquer que le pixel concerné est extérieur au VOP et la valeur 1 pour indiquer qu'il appartient au VOP. Son codage est identique au codage des BAB ;

2) une *carte à niveau de gris* qui contient les valeurs comprises entre 0 et 255. Chaque pixel a un niveau de transparence, fixé par sa valeur sur la carte à niveau de gris. Le codage de cette carte est un codage en texture (voir définition 7.2 page 167).

Les composantes *alpha* permettent d'atténuer les effets de la segmentation vidéo. En effet, lorsqu'une vidéo source est segmentée en VO pour être codée au format MPEG4, la technique de segmentation – qu'elle soit manuelle, automatique ou supervisée – est rarement parfaite.

Dans une image naturelle (une photographie), il est difficile de fixer précisément le contour des objets. Souvent, ceux-ci ont une précision en deçà du pixel.

La figure 7.18 illustre la composition d'une scène avec un arrière-plan et deux objets (les moulins) :

1) l'objet de droite est codé avec une composante *alpha* binaire. Le contour de l'objet apparaît très nettement ;

2) l'objet de gauche est codé avec une composante *alpha*. Les pixels au voisinage du contour offrent un dégradé de transparence qui améliore le fondu de l'objet dans l'arrière-plan.

Figure 7.18. *Simulation d'une scène avec à gauche un VO muni d'une composante* alpha *et à droite un VO muni d'une composante* alpha *binaire*

De plus, les composantes *alpha* peuvent également servir à des montages spéciaux comme celui montré en figure 7.19.

Classiquement, elles permettent d'effectuer le passage, en mode fondu, d'un plan-séquence à un autre plan-séquence ; la transition s'effectue en laissant apparaître de plus en plus nettement les images du nouveau plan-séquence en même temps que les dernières images du premier plan.

Figure 7.19. *Simulation d'une scène composée d'un arrière-plan et d'un VO dont la composante* alpha *est utilisée pour produire un effet spécial*

7.4.6.2. *Les panoramas*

Dans un plan-séquence (une prise de vue sans arrêter la caméra), certains éléments de la scène, tels que l'arrière-plan, restent partiellement, voire totalement, inchangés.

C'est le cas, par exemple, de la présentation d'un journal télévisé, de la présentation télévisuelle de la météo ou encore de l'enregistrement d'un match de tennis.

Ces situations offrent une structuration assez précise des scènes observées qui sont classiquement constituées d'un arrière-plan (relativement statique) et d'objets vidéo.

Partant de ce constat, le standard MPEG4 propose de coder un plan-séquence en plusieurs couches. Les VO sont codés suivant le profil choisi. En revanche, l'arrière-plan est codé comme étant une couche (VOL) à part. Son codage doit être optimal en termes de débit binaire et de temps de décodage.

En premier lieu, ceci signifie qu'un arrière-plan commun est construit pour toutes les images du plan-séquence. Un tel plan est appelé un *panorama* (*static sprite*). Il est la composition en une seule image de tous les arrière-plans du plan-séquence. Par construction même, un panorama peut être de taille supérieure à la taille des images de la vidéo source.

Par exemple :

– lors d'un zoom caméra, le panorama est l'arrière-plan de l'image obtenue quand le zoom est le plus faible (voir figure 7.20). Les objets sont alors perçus comme éloignés de la caméra ;

– lorsque la caméra effectue une translation perpendiculaire à la direction de vue, le panorama est construit en mettant bout à bout les arrière-plans de la séquence. La figure 7.21 illustre le procédé.

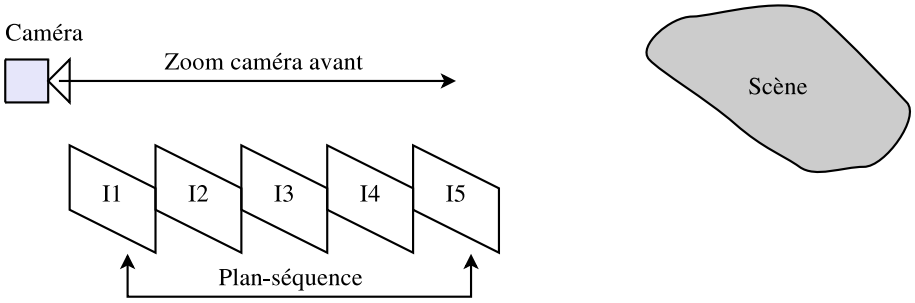


Figure 7.20. *Panorama d'un zoom caméra : l'arrière-plan de l'image I1 est choisi pour être le panorama*

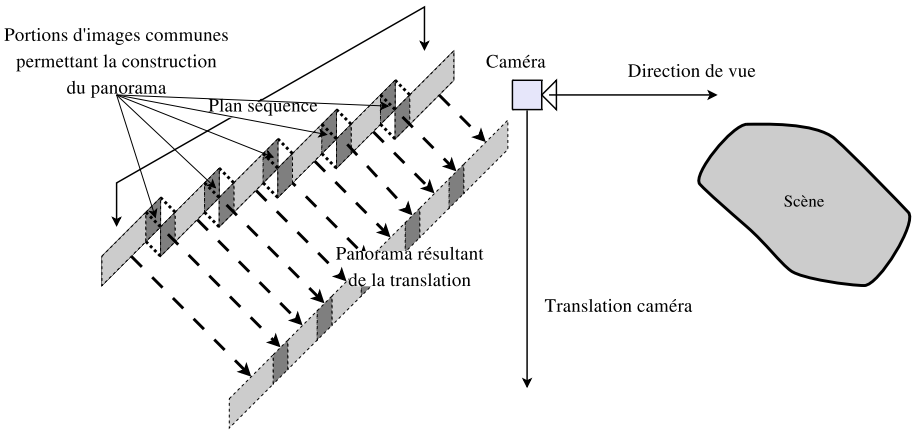


Figure 7.21. *Panorama d'une translation perpendiculaire à la direction de vue*

Le premier VOP du VOL est codé comme un I-VOP et les VOP suivants sont enregistrés comme des S-VOP (*Sprite VOP*). Suivant le débit binaire voulu ou admissible, deux modes de codage sont possibles :

1) le *mode basique* (*Basic Sprite*) code l'ensemble du panorama dans le I-VOP. Le décodeur récupère l'ensemble du panorama au décodage de ce I-VOP. Le panorama est rangé dans une mémoire dédiée afin d'être utilisé pendant tout le temps que le

plan-séquence est décodé. Chaque S-VOP, qui suit, contient de un à quatre vecteurs. Ces vecteurs informent le décodeur sur la transformation à appliquer au panorama pour qu'il corresponde à l'arrière-plan des VOP des autres VO du plan-séquence. Un seul vecteur indique une translation, deux ou trois vecteurs une transformation affine et quatre vecteurs une projection perspective ;

2) le *mode à faible latence* (*Low-Latency Sprite*) ne code que partiellement le panorama dans le I-VOP. Soit la totalité du panorama est codé mais à une qualité de résolution moindre que les autres objets. Soit une partie du panorama est codé. Dans les deux cas, la taille du panorama complet est également indiqué pour permettre au décodeur de le reconstruire. Les S-VOP contiennent les informations complémentaires. Soit il s'agit d'un nouveau morceau du panorama qu'il faut ajouter au panorama déjà décodé. Soit le S-VOP apportent les données résiduelles permettant d'augmenter la qualité de résolution du panorama déjà enregistré par le décodeur. Ces données résiduelles proviennent de la différence entre la haute résolution (la résolution initiale) et la basse résolution du panorama. Généralement, la basse résolution est obtenue par quantification. Ce mode évite les délais d'attente importants que l'on peut rencontrer avec le mode basique lorsque le réseau offre un faible débit et/ou que le panorama est de taille conséquente.

REMARQUE 7.2.– *Les panoramas qui viennent d'être décrits sont parfois dits statiques afin de ne pas les confondre avec les panoramas dynamiques (*dynamic sprite*) fournis par l'outil GMC (voir paragraphe 7.3.5).*

7.4.7. *Le profil* Advanced Coding Efficiency

Le profil ACE hérite du profil C auquel il ajoute (voir tableau 7.9) :

1) les composantes *alpha* ;

2) la vidéo entrelacée ;

3) la DCT adaptée aux formes ;

4) la prédiction au quart de pixel ;

5) la compensation de mouvement global.

Profil ACE :: Profil C
Composantes *alpha*
Vidéo entrelacée
DCT adaptée aux formes
Prédiction au quart de pixel
GMC

Tableau 7.9. *Les objets et outils du profil ACE*

7.4.7.1. *La DCT adaptée aux formes*

Avec le profil C, le codage en texture est effectué en utilisant une DCT 2D précédée, si besoin, de la technique de remplissage des macroblocs (voir paragraphes 7.3.1 et 7.4.2). Cette technique est conservée pour les macroblocs intérieurs tandis que les macroblocs extérieurs sont ignorés.

Mais pour les macroblocs contours, le profil ACE propose une nouvelle technique de codage en texture tenant compte de la forme des VOP à coder. Cette méthode de codage évite la technique de remplissage précédemment décrite.

Le principe repose sur une DCT adaptée aux formes (*shape-adaptative* DCT). Un jeu de DCT 1D (voir chapitre 3 volume 1) est utilisé au lieu d'une DCT 2D. Le traitement ne porte pas sur les macroblocs contours mais sur leurs blocs (de taille 8×8). L'algorithme est le suivant :

1) les valeurs des pixels intérieurs sont projetées sur la première ligne. Ainsi, chaque colonne voit ses pixels intérieurs rangés à partir de la première ligne (voir figure 7.22.b) ;

2) pour chaque colonne, la DCT 1D de longueur adéquate est appliquée. La longueur de la DCT correspond au nombre de pixels intérieurs que compte la colonne. La première ligne contient alors l'ensemble des coefficients DC de chacune des colonnes (voir figure 7.22c) ;

3) les valeurs non transparentes du bloc résultant sont projetées sur la première colonne (voir figure 7.22d) ;

4) la DCT 1D de longueur adéquate est appliquée mais sur chacune des lignes. La valeur située en première ligne, première colonne, est le coefficient DC de l'ensemble des pixels intérieurs du bloc initial (voir figure 7.22e).

Ensuite, un parcours en zigzag de type *(c)* est effectué (voir figure 7.6 page 168). Lors de ce réordonnancement, les pixels extérieurs sont ignorés. Enfin, le codage entropique enregistre les valeurs dans le flux binaire. Le décodeur retrouve les valeurs grâce au masque *alpha* binaire qui le renseigne sur les positions et le nombre de pixels intérieurs que contient chaque ligne et chaque colonne. Il effectue le procédé à l'inverse du codeur à l'aide d'un jeu de IDCT 1D.

7.4.8. *Le profil* N-Bit

Le profil Nb complète le profil C avec un outil qui permet de gérer le nombre de bits assignés par pixel. Au lieu de conserver le codage sur 8 bits utilisés par les autres profils, cet outil propose un codage des pixels allant de 4 à 12 bits. Cependant, le codage des composantes *alpha* reste sur 8 bits.

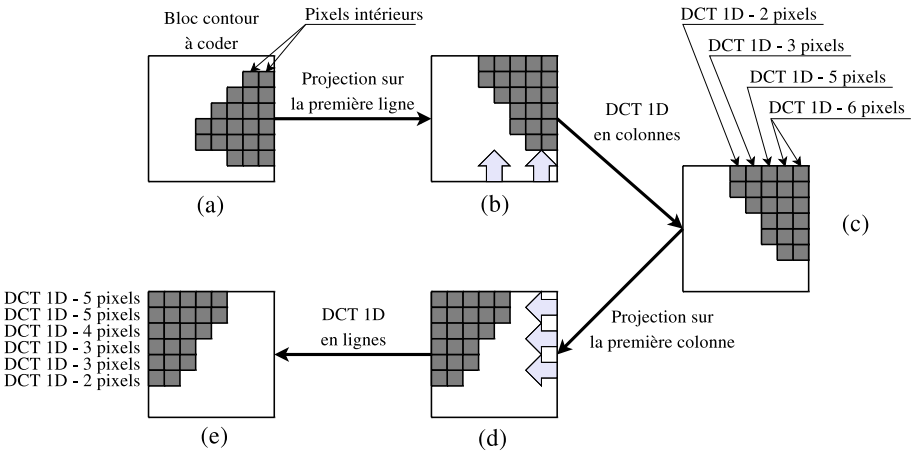


Figure 7.22. *DCT adaptée aux formes appliquée à un bloc contour*

7.5. Le codage granulaire

En plus, des granularités spatiale et temporelle telles que celles proposées par le standard MPEG2 (voir paragraphe 6.2.5), MPEG4 propose une *granularité binaire*. Celle-ci est particulièrement adaptée à la transmission sur des réseaux de largeurs de bandes inconnues. La vidéo source est découpée en un flux principal et un flux complémentaire. Sauf demande expresse du décodeur, le codeur envoie le flux principal et une *version tronquée* du flux complémentaire. Le niveau de la troncature est défini en fonction de la largeur de bande estimée du réseau.

Au niveau du codeur, le codage en texture (voir définition 7.2) – *intra* ou *inter* – est classique. Les coefficients quantifiés forment le flux principal. Parallèlement, ces coefficients sont déquantifiés et soustraits aux coefficients non quantifiés de la DCT. La différence obtenue, est codée en plans binaires pour former le flux complémentaire. Le procédé est décrit par la figure 7.23.

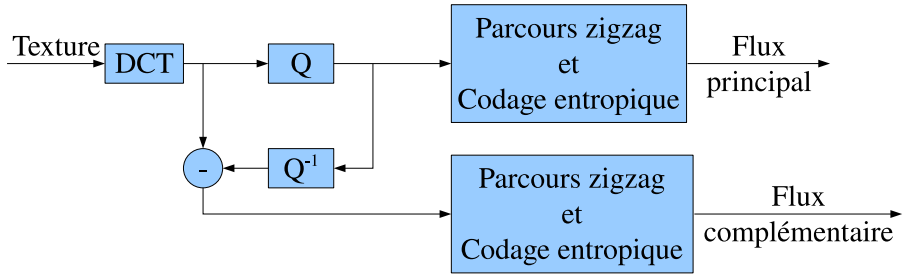


Figure 7.23. *Granularité binaire*

EXEMPLE 7.6.– *Soit la liste des différences suivantes obtenue après un parcours en zigzag :*

$$L = (+5, 0, 0, 0, -8, +13, 0, 0, 0 + 3, 0, \cdots, 0)$$

Les plans binaires des valeurs de cette listes sont présentés en tableau 7.10. Chaque plan est codé séparément des autres :

Plan 3 :	$4, 1(-), 0, 0(+), \mathrm{EOP}$
Plan 2 :	$0, 1(-), 4, \mathrm{EOP}$
Plan 1 :	$9, 0(+), \mathrm{EOP}$
Plan 0 :	$0, 4, 3, \mathrm{EOP}$

EOP *est la marque de fin de plan binaire (*End Of bitPlane*). Le signe est enregistré quand le bit de poids fort de la valeur est rencontré :* 0 *pour le signe* $+$ *et* 1 *pour le signe* $-$.

	-5	0	0	0	-8	+13	0	0	0	+3	0	$\cdots$	0
Signe	1	0	0	0	1	0	0	0	0	0	0	$\cdots$	0
Plan 3 (MBS)	0	0	0	0	1	1	0	0	0	0	0	$\cdots$	0
Plan 2	1	0	0	0	0	1	0	0	0	0	0	$\cdots$	0
Plan 1	0	0	0	0	0	0	0	0	0	1	0	$\cdots$	0
Plan 0 (LSB)	1	0	0	0	0	1	0	0	0	1	0	$\cdots$	0

Tableau 7.10. *Représentation en plans binaires de la liste* L

Chaque plan binaire codé est précédé d'un marqueur qui permet au codeur de tronquer le codage en plans binaires pour construire le flux complémentaire. Ces marqueurs sont également utilisés par le décodeur pour retrouver les différences tronquées.

EXEMPLE 7.7.– *Pour l'exemple précédent, avec une troncature au plan binaire 2, seuls les plans 2 et 3 sont codés par le codeur et décodés par le décodeur. Les autres plans sont mis à zéros. La liste* L *tronquée vaut alors :*

$$\hat{L} = (-4, 0, 0, 0, -8, +12, 0, \cdots, 0)$$

Ces différences tronquées sont ajoutées aux valeurs provenant du flux principal. Le décodeur fournit donc une présentation tronquée.

Il existe trois profils supportant le codage granulaire :

1) le profil *Simple Scalable* (SS) ajoute au profil *Simple*, les B-VOP et les granularités temporelle et spatiale pour les VOP de forme rectangulaire ;

2) le profil *Fine Granular Scalability* (FGS) hérite du profil *Simple* qu'il complète avec la quantification alternative, la vidéo entrelacée et la *granularité fine* pour les VOP de forme rectangulaire :

a) *la granularité fine spatiale* correspond à la ganularité binaire. On obtient un flux principal et un flux complémentaire tronqué,

b) *la granularité fine temporelle* est une combinaison de la granularité temporelle et de la granularité binaire. La granularité temporelle est utilisée pour créer un flux principal et un flux complémentaire. Le flux complémentaire est ensuite codé avec une granularité binaire (qui fournit à son tour deux flux). On obtient ainsi une granularité à trois niveaux.

Les VOP sont les images (ou trames) de la vidéo à coder ;

3) le profil *Core Scalable* (CS) reprend le profil *Core* auquel il ajoute la granularité temporelle pour les VOP de forme rectangulaire et la granularité spatiale pour les VOP de forme rectangulaire ou quelconque. Pour les VOP rectangulaires, les deux granularités peuvent être combinées pour générer un flux principal et deux flux complémentaires.

7.6. Le codage des images fixes

Suivant les applications, les images fixes sont des panoramas et/ou des images utilisées pour le placage de texture en synthèse d'images 2D et 3D (voir profil BAT en section 7.1). Elles sont, en général, de taille conséquente comparée à la taille des images d'une vidéo. Aussi, leur codage se fait à l'aide d'un outil spécifique *VTC Tool*. VTC est l'acronyme pour *Visual Texture Coding*, dont une traduction libre est : *codage de textures visuelles*.

Cet outil n'utilise pas la DCT mais la transformée en ondelettes discrète (DWT) qui fournit une analyse multirésolution (AMR). Les sous-bandes à différentes résolutions de l'AMR offrent une méthode directe et efficace de codage granulaire.

REMARQUE 7.3.– *Le chapitre 2 du volume 1 décrit l'AMR et fournit un exemple. La DWT a pour résultat une représentation hiérarchique de l'image à coder qui permet un contrôle précis, à la fois, du débit binaire et de la qualité (c'est-à-dire de la résolution spatiale). Cette technique est aussi utilisée par le standard JPEG2000 (voir chapitre 4).*

La figure 7.24 présente les sous-bandes, de l'exemple utilisé dans le chapitre 2 du volume 1, de manière à illustrer la forte corrélation qu'il y a entre les sous-bandes de détails verticaux (resp. diagonaux, horizontaux).

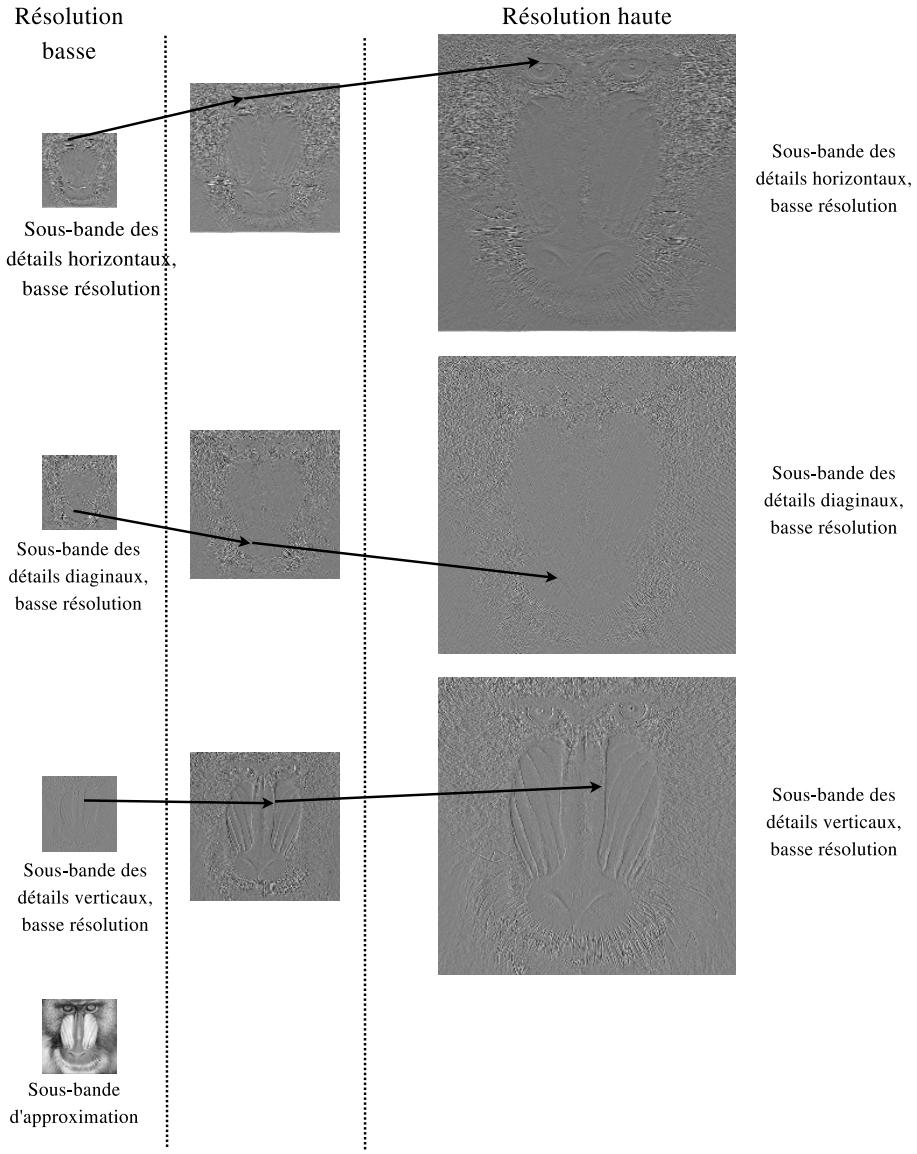


Figure 7.24. *Décomposition par une DWT (ondelettes de Haar). Les sous-bandes de détails sont fortement corrélées entre elles. Les flèches indiquent les positions similaires d'une sous-bande à l'autre*

7.6.1. *L'outil VTC*

L'outil VTC effectue une transformée en ondelettes discrète de l'image à l'aide des filtres de Daubechies 9/3. Les coefficients de la sous-bande d'approximation sont codés séparément des coefficients des sous-bandes des détails.

7.6.1.1. *Le codage de la sous-bande d'approximation*

La sous-bande d'approximation est codée spatialement. Chaque coefficient est quantifié avec un quantificateur uniforme. Puis, la différence avec son voisinage est envoyé à un codeur entropique de type arithmétique. Le voisinage est défini par les coefficients spatialement proches et déjà codés.

7.6.1.2. *Le codage des sous-bandes des détails*

Les sous-bandes de détails sont organisées hiérarchiquement. Les détails fins sont présents dans les sous-bandes de hautes résolutions. Les détails les plus prononcés[6] sont dans les sous-bandes voisines de la sous-bande d'approximation.

Cette hiérarchie est utilisée pour ordonner les coefficients des sous-bandes. L'ordonnancement peut se faire en respectant le niveau de résolution des sous-bandes ou en respectant l'arborescence des coefficients :

– *l'ordonnancement en sous-bandes*. L'ensemble des coefficients des sous-bandes de basse résolution est codé en premier. Puis, le procédé est répété pour la résolution supérieure et ainsi de suite jusqu'à la résolution maximale (voir figure 7.25) ;

– *l'ordonnancement arborescent*. Chaque coefficient des sous-bandes de la basse résolution est suivi des coefficients des sous-bandes des hautes résolutions qui sont situés à la même position (à un suréchantillonnage près). Cette relation est illustrée dans la figure 7.24 par les flèches. On obtient une forêt (arborescence à plusieurs racines) de coefficients. Chaque racine est un coefficient de la sous-bande des détails verticaux de la basse résolution. Les feuilles des arbres sont les coefficients des sous-bandes de la plus haute résolution. Cette forêt est codée à l'aide d'un seuil τ, et de quatre symboles [SHA 93] :

- *P*, le coefficient est supérieur au seuil τ. Il est significatif (c'est-à-dire supérieur au seuil τ en valeur absolue). Le symbole *P*ositif est codé,

- *N*, le coefficient est inférieur au seuil (négatif) $-\tau$. Il est significatif. Le symbole *N*égatif est codé,

- *I*, le coefficient est compris entre $-\tau$ et τ. Il est insignifiant. Mais certains de ses descendants sont significatifs. Le symbole *I* (valeur nulle isolée) est codé,

- *Z*, le coefficient et l'ensemble de ses descendants sont insignifiants. Le symbole *Z* (arbre à valeurs nulles) est codé et aucun descendant ne sera codé par la suite,

L'annexe D.2 fournit un exemple illustratif de ce codage arborescent.

6. Donc présents dans les basses et moyennes fréquences.

Cet ordonnancement est suivi ou précédé par une étape de quantification qui peut être :

1) une *quantification scalaire* qui applique un unique pas de quantification à tous les coefficients de la transformée en ondelettes (excepté les coefficients de la sous-bande d'approximation) ;

2) une *quantification multiple* qui utilise un jeu de pas de quantification allant de Q_0 à Q_n. Chaque coefficient est tout d'abord quantifié avec le pas de quantification Q_0 pour être ensuite codé avec l'algoritme EZW. Parallèlement, le coefficient quantifié est soustrait au coefficient initial. L'erreur de quantification due au pas Q_0 est, à son tour, quantifiée avec le pas de quantification Q_1 et codée avec l'algorithme EZW. Le procédé est répété jusqu'au pas de quantification Q_n, comme le montre la figure 7.26 ;

3) une *quantification en plans binaires* qui requiert que les coefficients soient réordonnés à l'aide de l'algorithme EZW. Ensuite, chaque plan binaire est codé séparément des autres (voir section 7.5).

7.6.2. *Les profils* Scalable Texture *et* Advanced Scalable Texture

Le profil ST propose l'outil VTC pour le codage des VO de forme rectangulaire uniquement. Le profil AST propose le codage d'images fixes de très grandes taille en opérant un découpage en sous-images (voir définition au chapitre 4). Chaque sous-image est codée à l'aide de l'outil VTC. La figure 7.27 montre le découpage en sous-images.

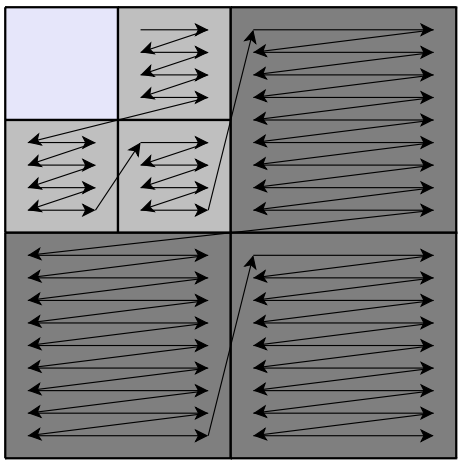


Figure 7.25. *Codage en sous-bandes*

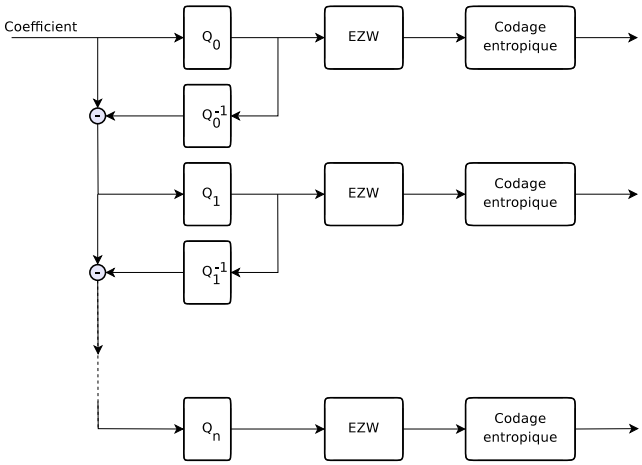


Figure 7.26. *La quantification multiple*

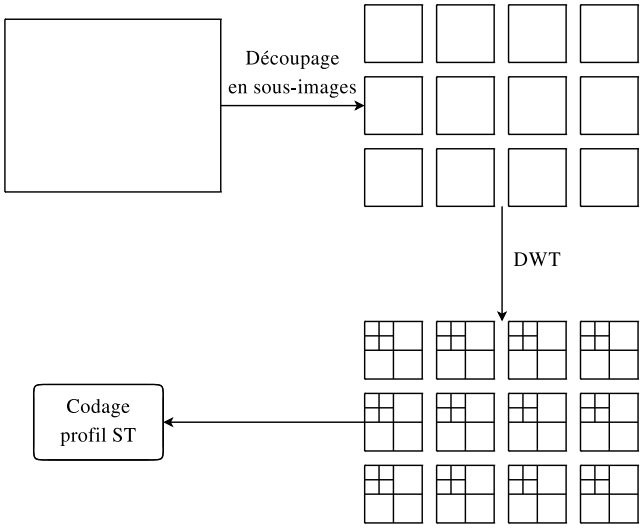


Figure 7.27. *Découpage d'une image en sous-images avant codage*

7.7. Le codage professionnel

Avant toute diffusion, un multimédia doit être conçu et mis en forme par une ou plusieurs équipes de production. Ces équipes interdisciplinaires regroupent des informaticiens, des spécialistes du signal et de l'électronique, des artistes, des ergonomes, des infographistes, etc.

La communication du multimédia entre ces équipes ne souffre aucune détérioration. Bien au contraire, il faut que la qualité des objets audioviduels (AVO) soit excellente pour permettre la modification ou l'ajout d'objets.

C'est dans cet objectif que le standard MPEG4 propose deux profils, dits *Studio*, de qualité supérieure aux autres profils. Cette augmentation de la qualité entraîne, évidemment, une diminution du taux de compression. Le débit doit donc se faire sur des supports à large bandes, totalement (ou quasi) insensibles aux pertes et aux perturbations que l'on rencontre sur les réseaux classiques (qui vont d'Internet à la téléphonie mobile). La communication de multimédias en phase de production est faite à l'aide de DVD *Blue-Ray*, de disques durs (partages d'espaces) ou encore de réseau locaux à (très) hauts débits.

Le codage est donc sans perte ou quasi sans perte avec un échantillonnage couleur de type 4:4:4 (pas de sous-échantillonnage) ou de type 4:2:2 si l'on veut une conversion vers un support au format MPEG2 (un DVD par exemple).

7.7.1. *Le profil* Simple Studio

Les caractéristiques du profil SS sont :

1) une vidéo source progressive ou entrelacée avec un échantillonnage couleur de type 4:4:4, 4:2:2 ou 4:2:0 ;

2) un codage en I-VOP uniquement ;

3) une transformée en cosinus discrète (DCT) faite avec une précision augmentée de trois bits par rapport aux autres profils. Ceci autorise un codage sans perte ;

4) un mode sans compression du codage. Le flux binaire est de taille identique ou inférieure à la taille de la source ;

5) un codage des composantes *alpha* binaires sans compression ;

6) un codage des composantes *alpha* entre 4 et 12 bits ;

7) reprise des bandes de macroblocs du standard MPEG1/2.

7.7.2. *Le profil* Core Studio

Le profil CS hérite du profil SS auquel il ajoute :

1) le codage des images fixes (voir section 7.6) ;

2) les P-VOP.

7.8. MPEG4 *Advanced Video Coding* (AVC)

La partie codage de vidéos avancé (AVC) du standard code uniquement des VOP de forme rectangulaire. Plus précisément, il code uniquement les images des vidéos, entrelacées ou progressives, au format 4:2:0 en *YCbCr*.

Le codage est donc proche du codage MPEG2. Les images sont découpées en macroblocs. Le codage des macroblocs est similaire au codage des macroblocs du format MPEG2.

Pour augmenter les taux de compression tout en gardant une qualité visuelle de type MPEG2, les macroblocs peuvent être découpés en blocs et les blocs en sous-blocs. De plus, le débit est géré par l'utilisation d'un simulateur de codage en tête du codeur. Ce simulateur indique, au codeur, le débit binaire et les paramètres associés. Le codeur peut alors effectuer un codage avec des taux élevés de compression.

Cette augmentation des taux de compression autorise un multiplexage plus fort (la bande passante, même faible, peut être partagée par plus de flux que ne l'autorise le format MPEG2).

Le multiplexage est également pris en compte par la séparation de la partie dédiée au codage de la vidéo : *Video Coding Layer* (VLC) ; de la partie organisation des données vidéo codées : *Network Abstraction Layer* (NAL). La figure 7.28 illustre le principe.

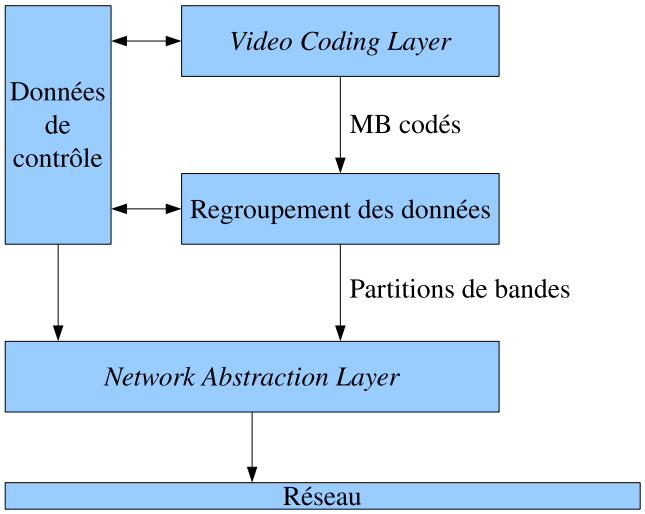


Figure 7.28. *Séparation du codage en une partie VLC et une partie NAL*

7.8.1. *Macroblocs et bandes*

Un macobloc MPEG4 est identique à un macrobloc MPEG2 mis à part qu'il est typé :

1) *I-MB*. Un I-MB est prédit en mode *intra* (paragraphe 7.8.4) à l'aide de MB codés (puis décodés) provenant de sa bande (donc de la même image). La prédiction peut se faire pour tout le macrobloc. On obtient alors un vecteur de prédiction (deux si la vidéo source est entrelacée). La prédiction peut également se faire sur chacun des blocs de taille 4×4 du macrobloc (voir figure 7.29). On obtient alors seize vecteurs (32 en mode entrelacé). Cette deuxième approche est choisie quand la prédiction pour l'ensemble du macrobloc comporte une erreur trop importante. Il existe une prédiction alternative I_PCM, décrite plus loin ;

2) *P-MB*. Un P-MB est prédit en mode *inter* (paragraphe 7.8.3) avec une image de référence provenant d'une liste intitulée *list0*. La prédiction peut se faire pour tout le macrobloc. On obtient un (ou deux) vecteur(s) de prédiction. Le macrobloc peut aussi être décomposé en blocs de taille 16×8, 8×16 ou 8×8 (voir figure 7.30). Chaque bloc est prédit à partir d'une image de la liste *list0*. Si la meilleure prédiction est faite avec les quatre blocs 8×8 du macroblocs, le procédé est répété sur chacun des blocs. Chaque bloc 8×8 est décomposé en *sous-blocs* de taille 8×4, 4×8 ou 4×4 (voir figure 7.30). Tous les sous-blocs d'un bloc sont prédits à partir de la même image de la liste *list0*. A nouveau la meilleure prédiction (8×8 ou 4×4) est choisie ;

3) *B-MB*. Un B-MB est prédit en mode *inter* (paragraphe 7.8.3). La prédiction peut être faite avec une ou deux images de référence comme pour les B-VOP (voir paragraphe 7.3.3). Une des images de référence provient de la liste *list0* et la deuxième d'une seconde liste intitulée *list1*. Le découpage d'un macrobloc en blocs et en sous-blocs est également utilisé.

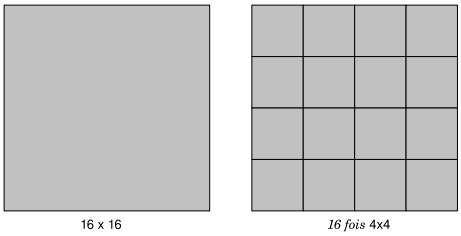


Figure 7.29. *Codage en* 16×16 *ou en* 4×4 *du mode* intra

Les prédictions des P-MB et des B-MB sont choisies dans les listes, *list0* et *list1*, d'images précédemment codées (et décodées). Le choix entre plusieurs images de référence potentielles permet de mieux prendre en compte les occultations, les changements de plans, etc. Ceci augmente encore la qualité de la prédiction.

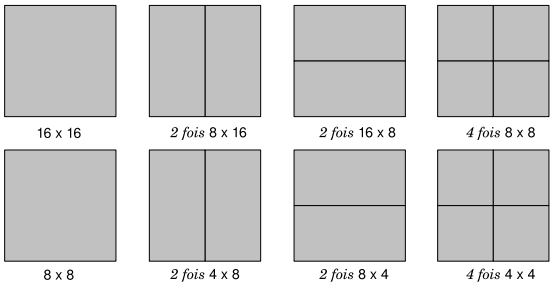


Figure 7.30. *Codage en* 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4 *du mode* inter

La terminologie de *bandes* (*slices*) de macroblocs (MB) utilisée par MPEG2 est reprise mais légèrement modifiée. Une bande regroupe des MB de même type :

1) une *I-bande* (*I-slice*, I-B) regroupe uniquement des macroblocs I-MB ;

2) une *P-bande* (*P-slice*, P-B) est constituée de I-MB et de P-MB ;

3) une *B-bande* (*B-slice*, B-B) contient des I-MB et des B-MB.

Les MB au sein d'une bande sont rangés dans l'ordre de codage, horizontal ou vertical[7]. Les bandes d'une image (ou d'une trame pour les vidéos entrelacées) sont organisées en *groupes de bandes* (GB). Si l'image ne comporte qu'un groupe alors les MB sont rangés suivante la technique utilisée par le standard MPEG2. Si l'image possède plusieurs groupes, ceux-ci peuvent être agencés suivant un des modèles suivants (voir figure 7.31) :

1) le modèle *entrelacé* alterne les groupes ;

2) le modèle *réparti* range les MB successifs dans différents groupes ;

3) le modèle *objet* forme des groupes pour chaque objet (région) de l'image plus un groupe dédié à l'arrière-plan ;

4) le modèle *escargot* découpe l'image en deux groupes dont un est centré dans l'image et sa taille est fixée par le codeur.

5) le modèle *horizontal* découpe l'image en deux parties qui se superposent. Les MB y sont rangés ligne par ligne ;

6) le modèle *vertical* découpe l'image en deux parties, gauche et droite. Les MBs y sont rangés colonne par colonne ;

7) le modèle *explicite* signifie que le codeur indique l'ordre qu'il souhaite utilisé (différent des six premiers) ; chaque MB possède alors une référence de groupe. La méthode de parcours est également définie par le codeur.

7. Parcours de l'image de haut en bas et de gauche à droite.

7.8.2. *Fonctionnement général*

Le schéma général du codeur est donné en figure 7.32. Le filtrage est effectué afin de supprimer les artéfacts de blocs dus à la quantification. La figure 3.10 (chapitre 3 page 70) montre une image où de tels artéfacts n'ont pas été traités.

Le filtre utilisé diminue les hautes fréquences afin d'atténuer les passages abrupts entre blocs. Ainsi les images de référence, une fois décodées, sont plus fidèles à leur originales et, de surcroît, les prédictions sont meilleures.

La partie AVC définit trois profils :

1) le profil *Baseline* utilisé pour des applications de type vidéotéléphonie, vidéo-conférence, communication sans fil (paragraphe 7.8.6) ;

2) le profil *Main* utilisé pour des applications de type diffusion TV, stockage vidéo (paragraphe 7.8.7) ;

3) le profil *eXtended* pour des applications multimédias à haute interactivité et composées d'objets hétérogènes (paragraphe 7.8.8).

Les trois paragraphes qui suivent présentent succinctement la prédiction *intra* (paragraphe 7.8.4) et la prédiction *inter* (paragraphe 7.8.3) et les étapes de transformation et de quantification (paragraphe 7.8.5).

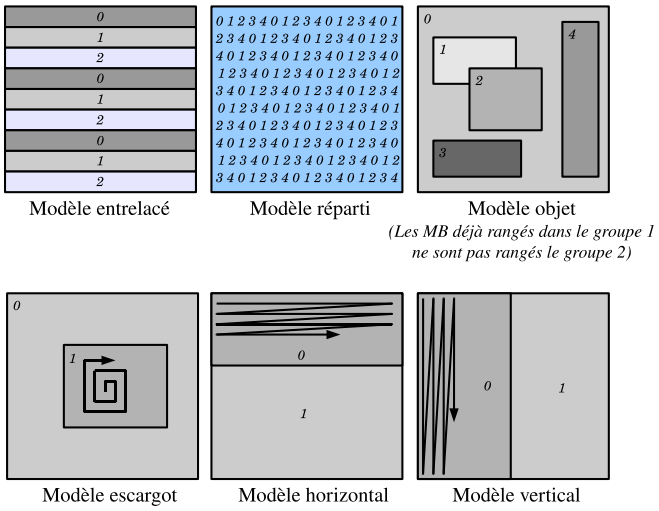


Figure 7.31. *Les modèles de groupes de bandes*

7.8.3. *Prédiction* inter

L'estimation de mouvement est faite au quart de pixel (voir annexe D.1). Compte tenu du partitionnement possible de chaque macrobloc, l'estimation fournit d'un à seize vecteurs par macrobloc (deux à 32 en vidéo entrelacée).

Le choix du partitionnement doit permettre de s'approcher au mieux de la courbe de la distortion en fonction du débit $D(R)$. C'est la même courbe qui sert de référence au standard JPEG2000.

Ce partitionnement en blocs et sous-blocs induit donc une compensation de mouvement arborescente optimale au sens de la courbe $D(R)$. Mais le nombre de vecteurs par macrobloc risque de devenir important et donc de pénaliser le débit *in fine*.

Pour éviter cette surcharge, les vecteurs de mouvement sont également prédits à partir des vecteurs des macroblocs, blocs ou sous-blocs voisins déjà codés. Le voisinage est défini comme le montre la figure 7.33. La prédiction respecte les règles suivantes :

1) pour les blocs de tailles 16×16, 8×8, 4×8, 8×4 et 4×4, la prédiction est la moyenne des voisins ;

2) pour les blocs de taille 16×8, la prédiction du bloc supérieur est le vecteur MV_B et celle du bloc inférieur est le vecteur MV_A ;

3) pour les blocs de taille 8×16, la prédiction du bloc gauche est le vecteur MV_A et celle du bloc droit est le vecteur MV_C ;

4) pour les macroblocs en mode silence (voir paragraphe 7.3.3.1), la prédiction est faite comme avec les macroblocs classiques (16×16).

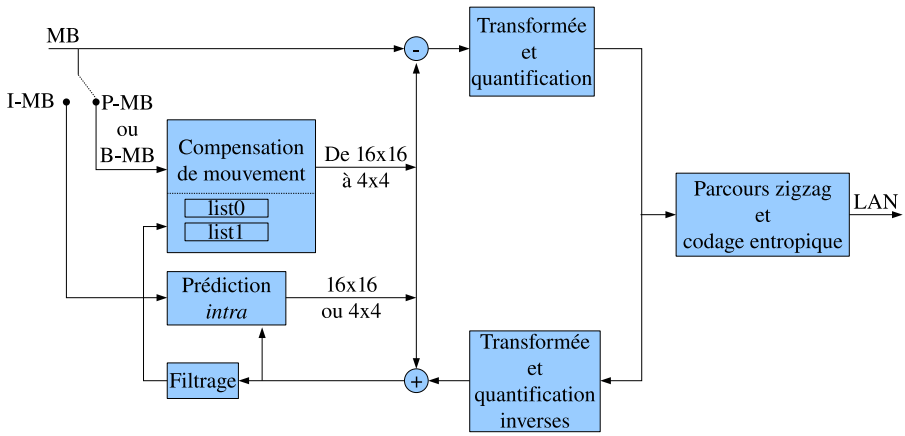


Figure 7.32. *Le codeur MPEG4-AVC*

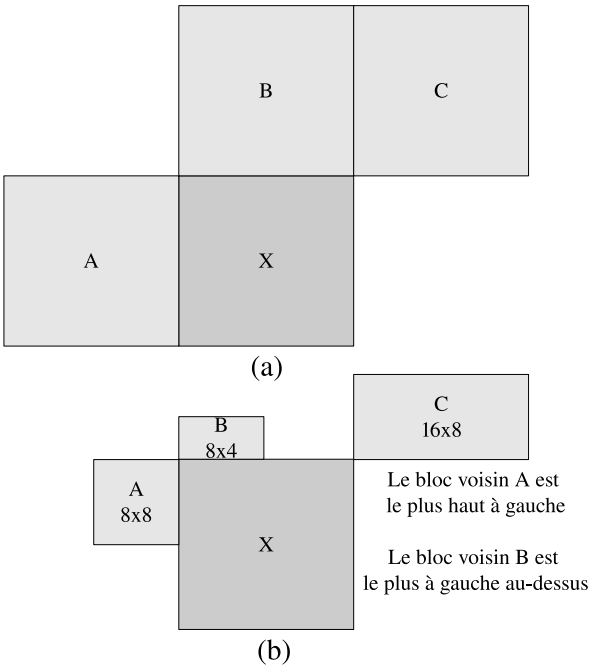


Figure 7.33. *(a) Le voisinage de X quand les voisins sont de même taille ; (b) le voisinage quand plusieurs blocs/sous-blocs bordent X*

7.8.4. *Prédiction* intra

La prédiction *intra* porte sur les macroblocs (taille 16×16) et les sous-blocs de taille 4×4 dans le domaine spatial – avant transformation contrairement à MPEG4 *Visual* (*Part 2*).

Il existe trois modes de prédiction : I_PCM, Intra_4×4 et Intra_16×16.

7.8.4.1. *Le mode I_PCM*

L'option I_PCM signifie qu'aucune prédiction et aucune transformation n'est appliquée au bloc en cours de codage. Les valeurs du bloc sont directement transmises au parcours en zigzag. Elle renseigne le codeur sur le nombre maximal de bits nécessaire pour coder le bloc.

7.8.4.2. *Le mode Intra_4×4*

Un bloc 4×4 utilise les pixels des blocs, déjà codés et reconstruits, qui lui sont à gauche et au-dessus, comme le montre la figure 7.34a.

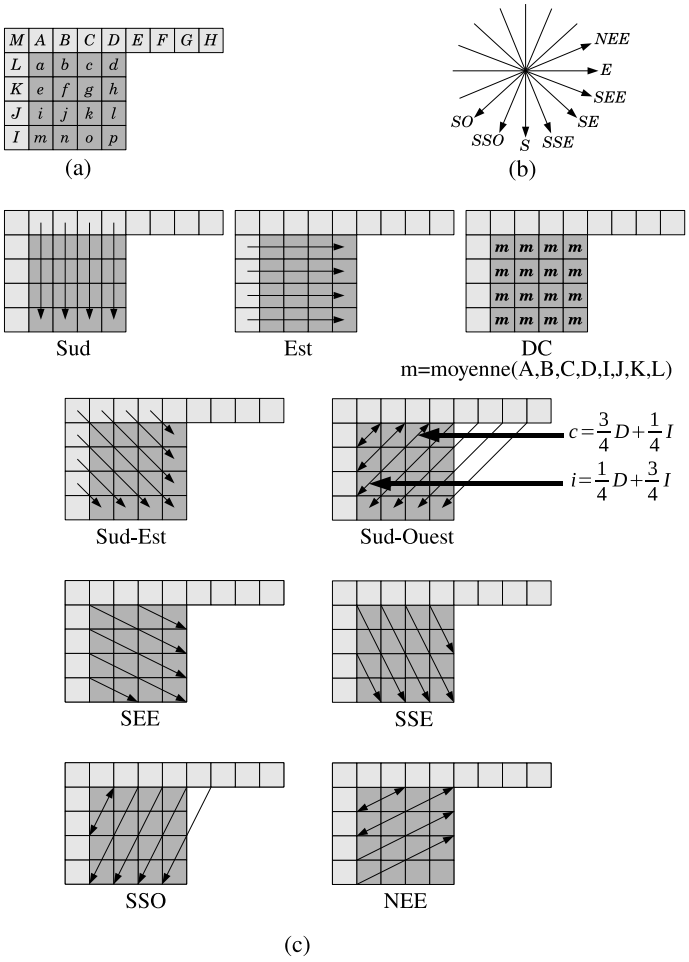


Figure 7.34. *Les neuf options de prédiction du mode Intra_*4×4

Il existe neuf options de prédiction décrites par la figure 7.34c (dont huit définies par les directions montrées en figure 7.34b) :

1) la prédiction *sud* recopie les pixels A, B, C et D sur chaque colonne ;

2) la prédiction *sst* recopie les pixels I, J, K et L sur chaque ligne ;

3) la prédiction *DC* recopie sur l'ensemble du bloc la moyenne des pixels $A \cdots D, I \cdots L$;

4) la prédiction *SE* recopie les pixels voisins suivant la direction sud-est ;

5) la prédiction *SO* recopie les pixels voisins suivant la direction sud-ouest ;

6) la prédiction *SEE* recopie les pixels voisins suivant la direction sud-est-est ;

7) la prédiction *SSE* recopie les pixels voisins suivant la direction sud-sud-est ;

8) la prédiction *SSO* recopie les pixels voisins suivant la direction sud-sud-ouest ;

9) la prédiction *NEE* recopie les pixels voisins suivant la direction nord-est-est.

7.8.4.3. *Le mode Intra_16×16*

La prédiction Intra_16×16 concerne les macroblocs. Elle est faite à partir des pixels immédiatement au-dessus et à gauche du macrobloc à coder. Il existe quatre options :

1) la prédiction *est* ;

2) la prédiction *sud* ;

3) la prédiction *DC* ;

4) la prédiction *sud-ouest*.

7.8.5. *Transformée et quantification*

Les blocs 4×4 sont transformés à l'aide d'une version adaptée de la DCT. Cette version est à coefficients entiers ce qui assure une transformation inverse exacte. La DCT est écrite sous sa forme matricielle 4×4 à laquelle certaines adaptations sont apportées [RIC 03, SAL 07]. La quantification est de la forme :

$$q = \left\lfloor \frac{x}{Q_{\text{step}}} \right\rfloor$$

où Q_{step} est le pas de quantification contrôlé par le paramètre QP, comme l'indique le tableau 7.11. Il y a au total 52 valeurs de pas possibles. Le paramètre QP permet d'adapter le pas à chaque bloc de taille 4×4.

Le mode Intra_4×4 est également utilisé pour les macroblocs. Les macroblocs sont alors découpés en blocs de taille 4×4. Chaque bloc est transformé à l'aide de la DCT adaptée, puis, quantifié. En revanche, les coefficients DC et AC quantifiés sont organisés hiérarchiquement (voir figure 7.35). Un premier bloc (de taille 4×4) regroupe les seize coefficients DC. Puis, suivent les blocs où les coefficients DC ont été mis à zéro.

QP	0	1	2	3	4	5	6	$\cdots$	51
Q_{step}	0,625	0,6875	0,8125	0,875	1	1,125	1,25	$\cdots$	224

Tableau 7.11. *Les pas de quantification Q_{step} en fonction du paramètre* QP

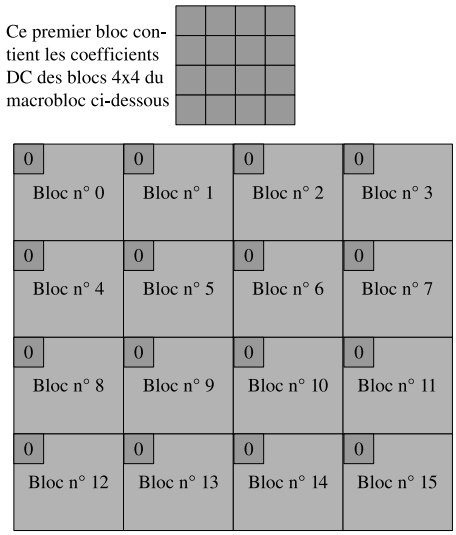


Figure 7.35. *Organisation hiérarchique des coefficients des blocs* 4×4 *d'un macrobloc*

7.8.6. *Le profil* Baseline

Le profil *Baseline* utilise les I-B et les P-B regroupées en GB. Ceci implique que seule la liste *list0* est utilisée pour la recherche des images de référence. Ce profil fonctionne également avec des *bandes redondantes*. Le tableau 7.12 synthétise les objets et outils de ce profil. Les prédictions sont *inter*, I_PCM, Intra_4×4 ou Intra_16×16. Les références, utilisées par l'estimation de mouvement, peuvent être faites en mode *rafraîchissement instantané du décodeur* (IDR). Le codage est de type CAVLC (sigle pour *Context-Based VLC*) :

1) avec une table prédéfinie pour les données résiduelles des blocs et des macroblocs ;

2) à l'aide de l'algorithme de Golomb pour toutes les données complémentaires.

7.8.6.1. *Les bandes redondantes*

Comme leur adjectif l'indique, les bandes redondantes servent de secours au décodeur quand celui-ci n'arrive pas à décoder correctement une image, ou une partie d'une image, à partir des seules I-bandes, P-bandes. De cette manière, le décodeur peut remplacer une zone endommagée d'une image. Si, en revanche, le décodage fonctionne normalement, le décodeur ignore les bandes redondantes. En particulier, il ne les stocke pas en mémoire.

Profil *Baseline*
I-B
P-B
GB
Bandes redondantes
IDR
Prédiction I-PCM
Prédiction Intra_4×4
Prédiction Intra_16×16

Tableau 7.12. *Les objets et outils du profil* Baseline

7.8.6.2. *Les images IDR*

Le codeur envoie une image étiquetée IDR quand il désire signaler un nettoyage de la liste des images de référence potentielles, *list0* et *list1*. Le décodeur en fait de même à la réception d'une telle image. Dans la pratique, les images des listes sont marquées *inutilisables*. Les P-B, à décoder par la suite, le sont sans référence. L'image étiquetée IDR est uniquement composée de I-B.

REMARQUE 7.4.– *La première image d'une séquence vidéo est toujours une image étiquetée IDR.*

7.8.7. *Le profil* Main

Le profil *Main* est similaire au profil *Baseline* auquel il soustrait les groupes de bandes (GB) et les bandes redondantes. Il ajoute les B-bandes (B-B), la vidéo entrelacée et la prédiction pondérée (voir tableau 7.13).

Profil M
I-B
P-B
B-B
Vidéo entrelacée
IDR
Prédiction I_PCM
Prédiction Intra_4×4
Prédiction Intra_16×16
Prédiction pondérée

Tableau 7.13. *Les objets et outils du profil* Main

Comme le montre le tableau 7.14, la prédiction se fait suivant le type de partition en blocs opéré sur le macrobloc en cours de codage. Pour les sous-blocs d'un bloc 8×8, le choix de la prédiction est celui fait pour le bloc 8×8.

Type de partition	Modes de prédiction
16×16	Avant (*list0*), arrière (*list1*), bidirectionnelle ou directe
16×8 ou 8×16	Avant (*list0*), arrière (*list1*) ou bidirectionnelle (Le choix est fait séparément pour chaque bloc du MB.)
8×8	Avant (*list0*), arrière (*list1*), bidirectionnelle ou directe (Le choix est fait séparément pour chaque bloc du MB.)

Tableau 7.14. *Les choix possibles de prédiction suivant le type de partition en blocs d'un macrobloc*

De plus, les prédictions des P-MB et des B-MB peuvent être *pondérées explicitement* ou *implicitement*. La pondération vient rééchelonner les valeurs des pixels du (ou des) macrobloc(s) de référence.

Quand elle est explicite, le poids w_0 (resp. les poids w_0 et w_1) est(sont) appliqué(s) aux valeurs des pixels du macrobloc de référence (resp. des macroblocs de référence). Le codeur transmet le(s) poids w_0 (et w_1) au décodeur.

Quand elle est implicite, la(les) pondération(s) est(sont) inversement proportionnelle(s) à la distance temporelle de la(des) référence(s) à l'image en cours de codage.

Une des applications de la prédiction temporelle, est d'ajuster les prédictions lors d'une transition d'un plan séquence au suivant. Pour une transition de type fusion, les dernières images du premier plan sont graduellement (dans le temps) recouvertes par les premières images du plan suivant.

7.8.8. *Le profil* eXtended

Le profil *eXtended*, aussi appelé profil X, hérite du profil *Baseline* auquel il ajoute les objets B-bandes (B-B), SP-Bandes (SP-B) et SI-bandes (SI-B) et les outils prédiction pondérée, codeur arithmétique binaire contextuel (CABAC) et regroupement des données (voir tableau 7.15).

7.8.8.1. *Les SP-bandes et SI-bandes*

MPEG4 a pour objectif (parmi d'autres) de transférer plusieurs flux vidéos. Il peut s'agir de plusieurs flux de différentes sources vidéo ou bien de plusieurs flux d'une

Profil X :: Profil *Baseline*
B-B
SP-B
SI-B
Prédiction pondérée
CABAC
Regroupement des données

Tableau 7.15. *Les objets et outils du profil X*

même source vidéo mais avec des débits binaires différents. Il faut donc introduire des outils et des objets qui permettent de passer d'un flux à l'autre de manière transparente et efficace (avec peu de délai d'attente).

EXEMPLE 7.8.– *Soit une vidéo codée en plusieurs flux avec différents débits binaires. A la réception de ces flux, le décodeur tente le décodage avec le débit le plus élevé. Mais, pour des raisons d'efficacité de son algorithme et/ou de qualité du réseau, il peut être amené à passer automatiquement à un flux ayant un débit binaire moindre.*

Si l'on considère deux flux vidéos, FV_1 et FV_2. Lors du décodage, le passage de FV_1 à FV_2 ne peut se faire que sous certaines garanties.

Pour des raisons de clarté, on suppose chaque image des deux flux constituée d'une seule P-bande (prédiction *inter* à une seule référence antérieure).

7.8.8.1.1. Introduire des I-bandes

La technique la plus simple consiste à enregistrer régulièrement une I-bande (une image codée I-B) pour éviter les prédictions et, donc, permettre le passage de FV_1 à FV_2 de manière transparente.

Bien qu'elle soit simple à mettre en œuvre, cette technique engendre un surcoût du débit binaire dus aux I-B régulièrement introduites.

7.8.8.1.2. Introduire des SP-bandes

Soient FV_1 et FV_2 deux flux d'une même source vidéo à différents débits binaires où des *points de passage* d'un flux (FV_1) à l'autre (FV_2) sont régulièrement enregistrés. Chaque point de passage est caractérisé par le codage de trois *SP-bandes* (voir figure 7.36) :

1) une SP-bande $\left(\mathrm{SP}_{(1),i}\right)$ fournit la différence entre l'image $P_{(1),i}$ de FV_1 et sa référence $P_{(1),j}(j < i)$;

2) une SP-bande $\left(\mathrm{SP}_{(2),i}\right)$ fournit la différence entre l'image $P_{(2),i}$ de FV_2 et sa référence $P_{(2),k}(k < i)$;

3) une SP-bande $\left(\mathrm{SP}_{(1,2),i}\right)$ fournit la différence entre l'image de référence $P_{(1),j}$ de FV_1 et l'image $P_{(2),i+1}$ de FV_2.

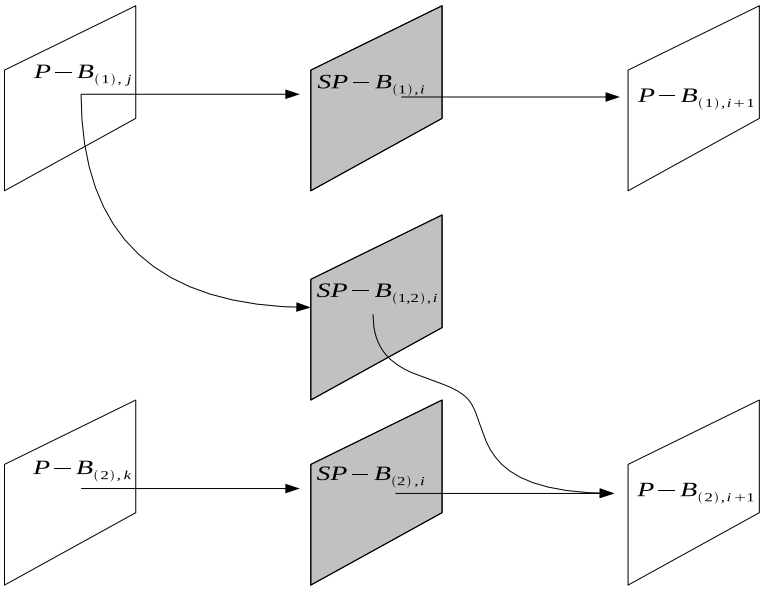


Figure 7.36. *Codage à l'aide de trois SP-bandes chaque point de passage*

Ainsi, on peut de manière transparente, décoder :

1) $P_{(1),i+1}$ à l'aide de la référence $P_{(1),j}$ et de $\mathrm{SP}_{(1),i}$;

2) $P_{(2),i+1}$ à l'aide de la référence $P_{(2),k}$ et de $\mathrm{SP}_{(2),i}$;

3) $P_{(2),i+1}$ à l'aide de la référence $P_{(1),j}$ et de $\mathrm{SP}_{(1,2),i}$.

Les trois SP-bandes sont prédites avec les techniques d'estimation et de compensation de mouvement décrites précédemment. Le débit binaire est alors moindre comparé à celui où des I-bandes sont régulièrement enregistrées comme points de passage.

REMARQUE 7.5.– *Les différences entre bandes sont calculées dans le domaine fréquentiel (après transformation par la DCT adaptée) et non dans le domaine spatial.*

Les SP-bandes peuvent également servir pour les accès directs. Pour une image n° i, le codeur génère une SP-bande $\mathrm{SP}_{(i,l)}$ qui codifie la différence entre l'image n° i et une image postérieure n° l ($l \gg i$). Ainsi, le décodage de l'image n° i et l'accès (*via* à un flux secondaire) à $\mathrm{SP}_{(i,l)}$ permet de reconstruire l'image n° l (voir figure 7.37).

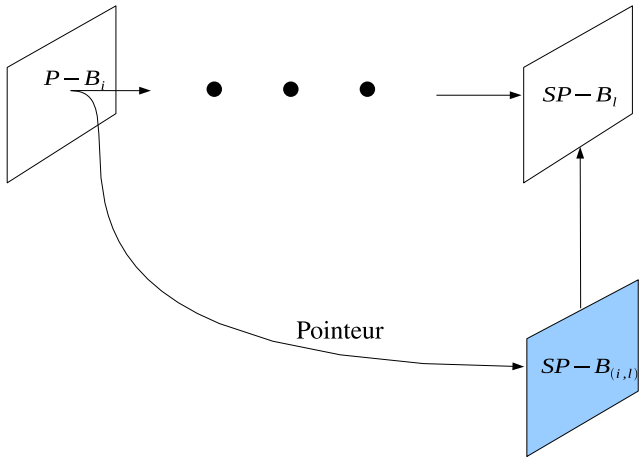


Figure 7.37. *Accès direct grâce aux SP-bandes*

7.8.8.1.3. Introduire des SI-bandes

Les *SI-bandes* sont similaires aux SP-bandes sauf que la prédiction Intra_4×4 est utilisée. Il n'y a donc pas d'estimation de mouvement.

Ce type de bandes permet de passer d'une séquence vidéo à une autre. Compte-tenu qu'il n'y a (sauf coïncidence) aucune corrélation entre les images des deux vidéos, la prédiction peut se faire sans compensation de mouvement.

7.8.8.2. *Le regroupement des données*

Cet outil sépare les données d'une bande en trois parties :

1) la première partie regroupe toutes les informations d'en-tête des macroblocs de la bande ;

2) la deuxième partie contient les données résiduelles (les différences) des macroblocs provenant des prédictions *intra* et des SI-B ;

3) la troisième partie rassemble les données résiduelles des MB provenants des prédictions en mode *inter* et des SPB.

La première partie doit être correctement transférée et décodée pour reconstruire la vidéo source. En revanche, sous certaines contraintes, le codeur peut ensuite choisir de décoder soit la deuxième partie, soit la troisième partie soit les deux. Le choix se fait suivant la bande passante du réseaux, l'algorithme de décodage utilisé et la mémoire disponible au sein du décodeur.

7.9. Synthèse

Un mot résume assez fortement l'apport du standard MPEG4 dans la définition, la structuration, du multimédia : l'*objet*.

Les vidéos, les sons sont découpés, segmentés en objets tels que chaque objet représente une information à forte sémantique. Par exemple, une vidéo d'un match de tennis peut être segmentée en trois objets vidéo : les deux joueurs et le terrain. Ceci permet au consommateur, au sens large, de remplacer, modifier ou substituer un des objets. Malheureusement, cette segmentation d'un audiovisuel en objets est encore du domaine de la recherche; bien des difficultés restent à surmonter avant de la voir devenir automatique en toutes circonstances.

Les informations et explications, fournies tout au long de ce chapitre, ont pour support le livre de I.E.G. Richardson [RIC 03] et celui de F. Pereira et T. Ebrahini [PER 02]. Pour des compléments d'information, en français, la revue *Technique de l'Ingénieur* propose des articles sur les standards MPEG principalement écrits par J.N. Gouyet et F. Mahieu [GOU 07].

Avec ce chapitre se termine cette introduction au multimédia. Mais, beaucoup d'éléments sont encore à explorer. MPEG4 a ouvert une voie vers l'interaction entre l'utilisateur et les objets multimédias que les normes MPEG7 et 21 viennent compléter. La nouvelle description du multimédia que propose le standard MPEG4 sera abordée de manière d'autant plus critique que les notions de base présentées dans ce livre sont connues dans leurs grandes lignes.

Chapitre 8

Conclusion

Les deux tomes de ce livre viennent de présenter le multimdia en trois parties. La première partie, consacrée aux théories liées au multimédia, est détaillée dans le premier tome. Les deux autres parties sont décrites dans le deuxième tome. La première partie de ce second tome est consacrée à la compression et à la représentation des images numériques alors que la deuxième partie est consacrée à la compression et à la représentation des audiovisuels numériques en terminant par la description de MPEG4. Ce standard est à la fois un standard actuel, avec sa partie MPEG4-AVC, et un ensemble d'objectifs dans divers domaines de la recherche.

Le fait que MPEG4 définisse tous les éléments qu'il manipule comme des objets, modifie fondamentalement la notion de multimédia. Ainsi, une vidéo devient une collection d'objets qui sont reliés entre eux par des contraintes spatio-temporelles; et, de même pour les audios. Il s'ensuit qu'un objet peut être naturel ou synthétique.

Cependant, découper une vidéo ou un audio en objets est encore (en 2008) du domaine de la recherche. Des résultats de cette recherche dépendra le succès de la *modélisation objet* (si, toutefois, on omet les habituelles contraintes financières).

Cette modélisation suppose qu'un processus a les mêmes capacités d'identification que l'être humain. Par exemple, dans une vidéo, un utilisateur peut facilement identifier les personnes et les objets (au sens usuel du terme), leurs dates d'arrivée et de disparition, leurs contraintes spatiales; le processus doit en faire de même.

C'est un véritable challenge qui, dans certaines circonstances, (reportages sportifs, par exemple) est réalisable dès aujourd'hui (en 2008). Mais, rien n'affirme que cette modélisation sera utilisable en toutes circonstances, avec, en prime, une complexité algorithmique et une complexité en mémoire peu élevées.

Aussi, d'autres techniques de modélisation sont proposées. Par exemple, la DCT 3D et les ondelettes 3D sont à l'étude pour fournir des flux vidéos de haute qualité faciles à constuire, exploitables sur des réseaux à débits faibles, variables et sensibles aux erreurs (Internet, téléphonie mobile).

De même, le codage entropique (sans perte) repose sur l'hypothèse d'échantillons indépendants. Mais quelle est la valeur d'entropie pour des échantillons dont la corrélation est modélisable ? Quels sont les conséquences sur les échantillonneurs et les capteurs numériques ? L'échantillonnage peut-il être irrégulier ? Dans ce cas, quels sont les algorithmes de reconstruction utilisables ?

A tous ces domaines de recherche, il faut ajouter le besoin *immédiat* de gérer la masse de plus en plus importante de supports multimédias. Les standards MPEG7 et MPEG21 ont été introduit dans ce sens et reprennent la notion d'objet en la modélisant au format XML pour l'indexation de données multimédias.

Il est donc fort probable que la modélisation objet devienne le *modèle générique* qui servira de communication entre les différents standards et normes de compression, d'indexation, etc.

Annexe A

Compléments : JPEG2000

A.1. Variables d'état

Lors de la phase Tier-1, chaque code-bloc $\mathcal{B}$ est codé indépendamment des autres codes-blocs plan binaire par plan binaire en commençant par les bits de poids fort. Pour ce faire, un échantillon $q[\mathbf{n}]$ est décrit par son signe $\chi[\mathbf{n}]$ et par son module $v[\mathbf{n}]$. La variable $v^p[\mathbf{n}]$ représente le bit du plan p du module $v[\mathbf{n}]$. p^{max} désigne le plan du bit de poids fort de ce module.

A cette représentation, s'ajoutent des variables d'état partagées par les trois procédures SPP, MRP et CUP, et l'algorithme général :

1) une variable de *détection* $\sigma[\mathbf{n}]$ est initialisée à 0 et prend la valeur 1 dès que le premier bit non nul de $v[\mathbf{n}]$ est codé. (Quand l'indice $\mathbf{n}$ est hors du code-bloc, sa valeur est considérée nulle) ;

2) une variable d'*affinage* $\gamma[\mathbf{n}]$ passe à 1 quand l'opérateur de raffinement du module (*magnitude refinement coding* MRC) vient d'être appliqué pour le bit du plan p. Sinon elle vaut 0 ;

3) une variable de *sélection* $\eta[\mathbf{n}]$ est initialisée à 0 à chaque nouveau plan p prêt à être codé. Il passe à 1 lorsque la valeur $v^p[\boldsymbol{n}]$ est susceptible de devenir significative. Autrement dit, $\eta[\mathbf{n}]$ passe à 1 dans la procédure SPP quand $v^p[\boldsymbol{n}]$ n'est pas encore significatif, mais, qu'au moins un de ses voisins l'est.

Chaque codage est représenté par une séquence de couples (κ, δ) qui indique le contexte (κ) et la décision (δ) associés au bit de position $\mathbf{n}$ dans le plan binaire p. Le contexte codifie l'état des voisins du bit observé.

A.2. Propagation des valeurs significatives : SPP

L'algorithme SIGNIFICANT_PROPAGATION_PASS parcourt les bandelettes colonne par colonne. Il recherche les bits qui ne sont pas encore significatifs, mais, dont au moins un voisin l'est : fonction VOISINS_SIGNIFICATIFS($\boldsymbol{n}$).

Si c'est le cas pour le bit en position $\boldsymbol{n}$, sa variable de sélection $\eta[\boldsymbol{n}]$, enregistre son passage en procédure SPP pour le plan binaire p. La procédure CODAGE_ZÉROS est alors appelée.

De plus, si le bit devient significatif, la procédure de codage du signe, CODAGE_SIGNE, est appelée à son tour et la variable de détection, $\sigma[\boldsymbol{n}]$, passe à 1.

```
SIGNIFICANT_PROPAGATION_PASS()
1  pour n ∈ B
2  faire si σ[n] = 0  et  VOISINS_SIGNIFICATIFS(n)
3         alors CODAGE_ZÉROS(n)
4               η[n] ← 1
5               si v^p[n] > 0
6                  alors CODAGE_SIGNE()
7                        σ[n] ← 1
```

```
VOISINS_SIGNIFICATIFS(n = [n1, n2])
1  retourner ( Σ_{j1,j2=-1}^{1} σ[n1 + j1, n2 + j2] ) ≠ 0 // σ[n] = 0
```

Dans l'algorithme CODAGE_ZÉROS($\boldsymbol{n}$), la fonction CONTEXTE_TABLE_CZ (H, V, D) retourne l'un des contextes définis par le tableau A.1 en fonction des voisins horizontaux (variable H), verticaux (variable V) et diagonaux (variable D) du bit en position $\boldsymbol{n}$. La décision δ, récupère la valeur du bit. La procédure MQ_ENCODE(κ, δ) génère le flux binaire compressé correspondant au couple (κ, δ). Cette procédure est détaillée au paragraphe 4.4.5.

```
CODAGE_ZÉROS(n)
1  H ← σ[n1 − 1, n2] + σ[n1 + 1, n2]
2  V ← σ[n1, n2 + 1] + σ[n1, n2 + 1]
3  D ← Σ_{k1,k2=±1} σ[n1 + k1, n2 + k2]
4  κ ← CONTEXTE_TABLE_CZ(H, V, D)
5  δ ← v^p[n]
6  MQ_ENCODE(κ, δ)
```

Contextes	LL et LH			HL			HH	
κ	H	V	D	H	V	D	D	$H+V$
0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	1
2	0	0	≥ 2	0	0	≥ 2	0	≥ 2
3	0	1	x	1	0	x	1	0
4	0	2	x	2	0	x	1	1
5	1	0	0	0	1	0	1	≥ 2
6	1	0	≥ 1	0	1	≥ 1	2	0
7	1	≥ 1	x	≥ 1	1	x	2	≥ 1
8	2	x	x	x	2	x	≥ 3	x

Tableau A.1. *Table de codage des zéros : x exprime « n'importe quelle valeur ». Suivant le type de sous-bande en cours de codage (LL, LH, HL ou HH), le contexte κ correspond à une certaine configuration du voisinage (H, V et D) du bit en position $\boldsymbol{n}$*

Dans la procédure CODAGE_SIGNE($\boldsymbol{n}$), les valeurs assignées aux variables H et V indiquent trois contextes de signes possibles :

1) le *contexte 0*. Les deux voisins sont non-significatifs ou sont significatifs, mais, de signes opposés ;

2) le *contexte +1*. un des voisins, ou les deux, sont significatifs et positifs ;

3) le *contexte −1*. un des voisins, ou les deux, sont significatifs et négatifs.

La fonction ETATS_TABLE_CS(H, V) retourne le couple $(\kappa, \hat{\chi})$ définie par le tableau A.2 où $\hat{\chi}$ correspond au codage des signes des voisins du bit en position $\boldsymbol{n}$.

```
CODAGE_SIGNE(n)
1  H ← χ[n1 − 1, n2]σ[n1 − 1, n2] + χ[n1 + 1, n2]σ[n1 + 1, n2]
2  V ← χ[n1, n2 − 1]σ[n1, n2 − 1] + χ[n1, n2 + 1]σ[n1, n2 + 1]
3  H ← sign(H) · min{1, |H|}
4  V ← sign(V) · min{1, |V|}
5  (κ, χ̂) ← ETATS_TABLE_CS(H, V)
6  si χ̂ · χ[n] = 1
7     alors δ ← 0
8     sinon δ ← 1MQ_ENCODE(κ, δ)
```

L'instruction conditionnelle (instruction n°6) vérifie que le signe de la valeur courante est positif et qu'une des trois conditions suivantes est vérifiée :

– au moins un des voisins horizontaux est significatif et positif ;

– les deux voisins horizontaux sont non significatifs ;

– les deux voisins horizontaux sont de signes opposés.

Si la condition est vérifiée, δ prend la valeur 0, sinon elle prend la valeur 1. Ainsi, le signe codé tient également compte du contexte.

H	V	$\hat{\chi}$	κ
1	1	1	13
1	0	1	12
1	−1	1	11
0	1	1	10
0	0	1	9
0	−1	−1	10
−1	1	−1	11
−1	0	−1	12
−1	−1	−1	13

Tableau A.2. *Table de codage des signes*

A.3. Affinage du module : MRP

La procédure MAGNITUDE_REFINMENT_PASS sélectionne les bits du plan p dont les modules $v[\mathbf{n}]$ ont été identifiés significatifs lors du traitement d'un plan binaire $q > p$:

$$\sigma[\mathbf{n}] = 1 \text{ et } \eta[\mathbf{n}] = 0$$

A l'aide du tableau A.3, la fonction CONTEXTE_TABLE_RC(HVD) range les bits sélectionnés en trois catégories :

1) $\kappa = 16$. Les bits dont le module a déjà été traité par cette même procédure, mais, pour un plan binaire de poids q supérieur à p ($\gamma[\mathbf{n}] = 1$) ;

2) $\kappa = 15$. Les bits qui n'ont pas encore subi d'affinage ($\gamma[\mathbf{n}] = 0$), mais, dont au moins un voisin direct est significatif ;

3) $\kappa = 14$. les bits qui n'ont pas encore subi d'affinage ($\gamma[\mathbf{n}] = 0$), mais, dont aucun voisin direct n'est significatif.

$\gamma[\mathbf{n}]$	HVD	κ
1	x	16
0	> 0	15
0	0	14

Tableau A.3. *Table de codage de l'affinage*

La décision δ prend la valeur du bit $v^p[\mathbf{n}]$ quel que soit le cas de figure.

```
MAGNITUDE_REFINMENT_PASS()
 1  pour n ∈ B
 2  faire si σ[n] = 1 et η[n] = 0
 3        alors H ← σ[n1 − 1, n2] + σ[n1 + 1, n2]
 4              V ← σ[n1, n2 − 1] + σ[n1, n2 + 1]
 5              D ← Σ_{k1,k2=±1} σ[n1 + k1, n2 + k2]
 6              HVD ← H + V + D
 7              κ ← CONTEXTE_TABLE_RC(HVD)
 8              δ ← v^p[n]
 9              MQ_ENCODE(κ, δ)
10              γ[n] ← 1
```

A.4. Nettoyage : CUP

La procédure CLEAN_UP_PASS fait appel à CODAGE_RÉPÉTITIONS($\mathbf{n}$) pour coder une colonne constituée de 4 bits non significatifs dont les voisins directs sont également non significatifs. Si, le bit est à 1, il devient significatif et son signe est codé à l'aide de la procédure CODAGE_SIGNE.

```
CLEAN_UP_PASS()
 1  pour n = [n1, n2] ∈ B
 2  faire si σ[n] = 0 et η[n] = 0
 3        alors si n2 mod 4 ≠ 0 ou COLONNES_SIGNIFICATIVES(n)
 4                alors CODAGE_ZÉROS(n)
 5                sinon CODAGE_RÉPÉTITIONS(n)
 6              si v^p[n] = 1
 7                alors CODAGE_SIGNE()
 8                      σ[n] ← 1
```

```
COLONNES_SIGNIFICATIVES(n)
1  retourner (Σ_{j1=−1}^{1} Σ_{j2=−1}^{4} σ[n1 + j1, n2 + j2]) ≠ 0
```

La procédure CODAGE_RÉPÉTITIONS est une version modifiée du *Run Length Coding* classique. Au sein d'une colonne de quatre bits, si le premier bit valant 1 est :

– en deuxième position, alors son code est $(0, 1)$;

– en troisième position, alors son code est $(1, 0)$;

– en quatrième position, alors son code est $(1, 1)$.

Puis le pointeur $\mathbf{n}$ est incrémenté pour se trouver à la prochaine place non encore codée.

REMARQUE A.1.– *Dans toutes les autres procédures, l'incrémentation de ce pointeur est omise, puisque ce dernier passe simplement au prochain échantillon dans la bandelette courante. (Si la position courante est le dernier échantillon de la bandelette, le processus passe à la bandelette suivante jusqu'à la fin du bloc.)*

```
CODAGE_RÉPÉTITIONS(n = [n_1, n_2])
 1  si (Σ_{j_2=1}^{3} σ[n_1, j_2]) = 0
 2     alors κ ← 17
 3           δ ← 0
 4           MQ_ENCODE(κ, δ)
 5     sinon κ ← 17
 6           δ ← 1
 7           MQ_ENCODE(κ, δ)
 8           κ ← 18
 9           si v^p[n_1, 1] = 1
10              alors δ ← 0
11                    MQ_ENCODE(κ, δ)
12                    δ ← 1
13                    MQ_ENCODE(κ, δ)
14                    n_2 ← 2
15              sinonsi v^p[n_1, 2] = 1
16                       alors δ ← 1
17                             MQ_ENCODE(κ, δ)
18                             δ ← 0
19                             MQ_ENCODE(κ, δ)
20                             n_2 ← 3
21
22                       sinon δ ← 1
23                             MQ_ENCODE(κ, δ)δ ← 1
24                             MQ_ENCODE(κ, δ)
25                             n ← (n_1 + 1, 0)^T
```

Annexe B

Complément : MPEG1

B.1. Les critères de dissimilitude

Lors de la recherche du macrobloc de référence, B_{ref}, devant correspondre au macrobloc à coder B, différents critères de dissimilitude peuvent être utilisés :

– la *moyenne des différences absolues* (MAD[1])

– la *moyenne des différences quadratiques* (MSD[2])

– la *fonction de corrélation* (CCF [3])

– la *fonction de classification des différences absolues* (PDC[4])

B.1.1. *La moyenne des différences absolues*

La moyenne des différences absolues (MAD) correspond à :

$$(\hat{d}_x, \hat{d}_y) = \underset{\substack{|d_x| \leqslant p, \\ |d_y| \leqslant p}}{\operatorname{argmin}} \ \mathrm{MAD}(B, Z_{\mathrm{ref}}, y_B + d_y, x_B + d_x)$$

1. *Mean-Absolute Difference*.
2. *Mean-Squared Difference*.
3. *Cross-Correlation Function*.
4. *Pixel Difference Classification*.

avec :

$$\mathrm{MAD}(B, Z_{\mathrm{ref}}, y_0, x_0) = \sum_{x=-\frac{n}{2}}^{\frac{n}{2}} \sum_{y=-\frac{m}{2}}^{\frac{m}{2}} |B(y,x) - B_{\mathrm{ref}}(y+y_0, x+x_0)|$$

où :

– B est le macrobloc à coder de taille $m \times n$ de centre (y_B, x_B),
– Z_{Ref} l'image de référence dans laquelle le macrobloc est recherché,
– (d_x, d_y) le vecteur de déplacement tel que $(d_x, d_y) \in \{-p, +p\}^2$.

Pour $n = m = 16$ et $p = 6$:

$$(\hat{d}_x, \hat{d}_y) = \underset{\substack{|d_x| \leqslant 6,\\ |d_y| \leqslant 6}}{\operatorname{argmin}} \sum_{x=-8}^{8} \sum_{y=-8}^{8} |B(y,x) - Z_{\mathrm{ref}}(y + x_B + d_y, x + x_B + d_x)|$$

B.1.2. *La moyenne des différences quadratiques*

La moyenne des différences quadratiques (MSD) :

$$(\hat{d}_x, \hat{d}_y) = \underset{\substack{|d_x| \leqslant p,\\ |d_y| \leqslant p}}{\operatorname{argmin}} \mathrm{MSD}(B, Z_{\mathrm{ref}}, y_B + d_y, x_B + d_x)$$

avec :

$$\mathrm{MSD}(B, Z_{\mathrm{ref}}, y_0, x_0) = \sum_x \sum_y (B(y,x) - Z_{\mathrm{ref}}(y + y_0, x + x_0))^2$$

B.1.3. *La fonction de corrélation*

La fonction de corrélation (CCF) vaut :

$$(\hat{d}_x, \hat{d}_y) = \underset{\substack{|d_x| \leqslant p,\\ |d_y| \leqslant p}}{\operatorname{argmin}} \mathrm{CCF}(B, Z_{\mathrm{ref}}, y_B + d_y, x_B + d_x)$$

avec :

$$\mathrm{CCF}(B, Z_{\mathrm{ref}}, y_0, x_0) = \frac{\sum_x \sum_y B(y,x) Z_{\mathrm{ref}}(y + y_0, x + x_0)}{\sqrt{B^2}\sqrt{Z_{\mathrm{ref}}^2}}$$

et :

$$\begin{aligned} B^2 &= \textstyle\sum_x \sum_y B^2(y,x) \\ Z_{\mathrm{ref}}^2 &= \textstyle\sum_x \sum_y Z_{\mathrm{ref}}^2(y + y_0, x + x_0) \end{aligned}$$

B.1.4. *la fonction de classification des différences absolues*

La fonction de classification des différences absolues (PDC) :

$$\begin{array}{rcll}(\hat{d}_x, \hat{d}_y) & = & \underset{\substack{|d_x| \leqslant p,\\ |d_y| \leqslant p}}{\operatorname{argmax}} & \mathrm{PDC}(B, Z_{\mathrm{ref}}, y_B + d_y, x_B + d_x) \\ & = & \underset{\substack{|d_x| \leqslant p,\\ |d_y| \leqslant p}}{\operatorname{argmax}} & \sum_x \sum_y T(B, Z_{\mathrm{ref}}, y_B + d_y, x_B + d_x, y, x)\end{array}$$

avec :

$$T(B, Z_{\mathrm{ref}}, y_c, x_c, y, x) = \begin{cases} 1 & \text{si } |B(y,x) - Z_{\mathrm{ref}}(y + d_y, x + d_x)| < \tau \\ 0 & \text{sinon} \end{cases}$$

où τ est un seuil prédéfini.

B.1.5. *Comparaison*

Bien que moins performante que la MSD ou que la CCF, la fonction MAD est préférée pour sa relative efficacité en temps de calcul et sa simplicité de mise en œuvre. Il est à noter que la fonction de classification PDC est à maximiser.

EXEMPLE B.1.– *Soit un macrobloc de taille* 2×2 *comme celui de la figure B.1.b et une zone de recherche dans l'image de référence de paramètre* $p = 2$ *(voir figure B.1a). Le critère de dissimilitude utilisé est la fonction MAD. Il y a* $(2p+1)^2 = 25$ *évaluations du critère dont les résultats sont présentés en figure B.1c. Le minimum est indiqué en gras sur la figure. L'estimation du mouvement fournit :*

1) le déplacement $(\hat{d}_x, \hat{d}_y) = (0, -2)$ *;*

2) le macrobloc de référence :

$$\boxed{\begin{matrix} 35 & 37 \\ 35 & 35 \end{matrix}}$$

B.2. L'algorithme de recherche du macrobloc de référence

Muni d'un critère de dissimilitude, la recherche du macrobloc de référence peut se faire de manière exhaustive. Toutefois, le coût d'une telle recherche étant prohibitif, des techniques sous-optimales, mais efficaces en temps de calcul, sont proposées. Deux parmi toutes celles qui existent sont décrites dans cette section : la *recherche en trois sauts* et la *recherche logarithmique*.

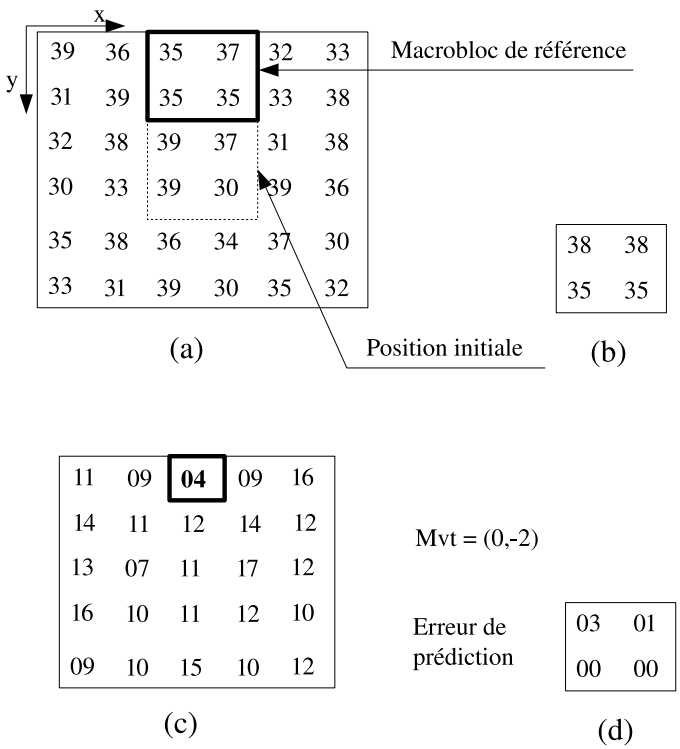


Figure B.1. *Estimation du mouvement : (a) zone de recherche dans l'image de référence, (b) macrobloc à coder, (c) évaluation du critère MAD avec son minimum encadré en gras et (d) le vecteur de déplacement et la différence entre les macroblocs source et référence*

B.2.1. *Recherche en trois sauts*

Cet algorithme procède en trois phases comme l'indique son nom :

– la première étape évalue le critère au centre de la zone de recherche (c'est-à-dire la position (y_B, x_B) du bloc à coder) et sur huit positions voisines, comme le montre la figure B.2. Ces positions sont à équidistance du centre et de la limite de la zone de recherche ; la position fournissant le coût minimal devient le nouveau centre ;

– la deuxième étape consiste à évaluer le critère sur ce nouveau centre et sur les huit voisins situés à une distance moitié de celle utilisée pour les précédents voisins; la position ayant le coût minimal est sélectionnée ;

– le processus est itéré jusqu'à ce que les voisins soient à une distance de 1 du centre ou quand le centre a le coût minimal; à la dernière étape, la position offrant le coût minimal sera la candidate à la compensation de mouvement.

EXEMPLE B.2.– *Le procédé est développé sur un exemple illustré en figure B.2:*

1) les valeurs du critère MAD pour le centre et ses huit voisins (étiquetés par le nombre 1) sont calculées; les voisins sont situés à mi-chemin entre le centre et le bord de la zone de recherche : la distance vaut 4 ;

2) la valeur minimum est supposée être en $(-4, -4)$*; cette position devient le nouveau centre et la distance est divisée de moitié ;*

3) le critère MAD est évalué pour les huits nouveaux voisins (étiquetés avec le nombre 2); la distance vaut 2 ;

4) Le minimum est en $(-2, -6)$ *;*

5) le critère pour les huits voisins (étiquetés avec le nombre 3) est évalué; la distance vaut 1 ;

6) Le minimum final (distance de 1) est en $(-3, -7)$.

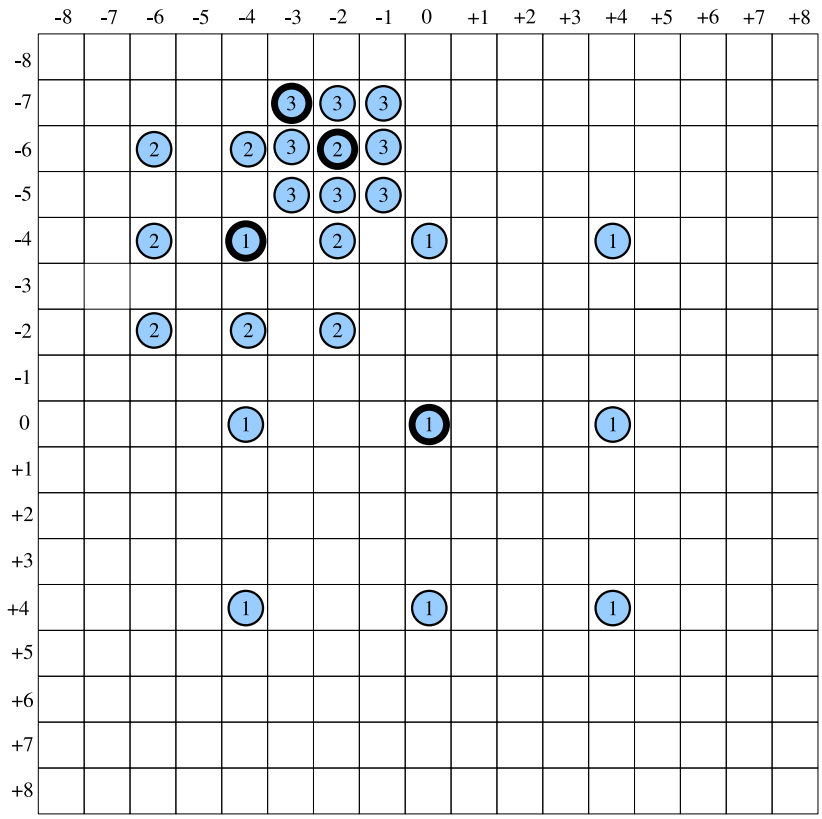


Figure B.2. *Algorithme de recherche trois-pas avec* $p = 8$

```
[x, y] = RECHERCHE_EN_TROIS_SAUTS(B, Z_ref, y_B, x_B, p)
 1  // (x_B, y_B) sont les coordonnées du centre du bloc B dans la P-image
 2  // p est la distance de recherche dans l'image de référence Z_ref
 3  p_x ← ⌊n/2⌋ + p
 4  p_y ← ⌊m/2⌋ + p
 5  d_y ← ⌊p/2 + n/2⌋
 6  d_x ← ⌊p/2 + m/2⌋
 7  x ← x_B
 8  y ← y_B
 9  OK ← vrai
10  V_min ← MAD(B, Z_ref, y, x)
11  tant que d ⩾ 1 AND OK = vrai
12  faire // recherche dans la zone de référence
13        x_c ← x
14        y_c ← y
15        pour k = 1 à 8
16        faire parmi
17              cas k = 1 :
18                  x' ← x + d_x ; y' ← y
19              cas k = 2 :
20                  x' ← x + d_x ; y' ← y − d_y
21              cas k = 3 :
22                  x' ← x ; y' ← y − d_y
23              cas k = 4 :
24                  x' ← x − d_x ; y' ← y − d_y
25              cas k = 5 :
26                  x' ← x − d_x ; y' ← y
27              cas k = 6 :
28                  x' ← x − d_x ; y' ← y + d_y
29              cas k = 7 :
30                  x' ← x ; y' ← y + d_y
31              cas k = 8 :
32                  x' ← x + d_x ; y' ← y + d_y
33            si |x' − x_B| ⩽ p_x AND |y' − y_B| ⩽ p_y
34              alors V' ← MAD(B, Zref, y', x')
35                    si V' < V_min
36                      alors V_min ← V'
37                            x ← x' ; y ← y'
38        si x_c = x AND y_c = y
39          alors OK ← faux
40        d_x ← ⌊d_x/2⌋
41        d_y ← ⌊d_y/2⌋
```

B.2.2. *Recherche logarithmique*

Tant que la distance de voisinage est supérieure à 1, les voisins horizontaux et verticaux sont consultés. La distance de voisinage est réduite de moitié uniquement quand le centre a la valeur minimale. Lorsque cette distance est de 1, les huit voisins sont consultés. Celui qui a la valeur minimale est alors le bloc recherché.

EXEMPLE B.3.– *La figure B.3 illustre le procédé :*

1) le centre et ses voisins (n°1) sont évalués ;

2) le minimum est en $(0, -4)$ *;*

3) le centre $(0, -4)$ *et ses voisins (n°2) sont évalués ;*

4) le nouveau centre $(-4, -4)$ *et ses voisins (n°3) sont évalués ;*

5) le centre est le minimum, donc la distance est diminuée de moitié (d=2) ;

6) le centre et ses nouveaux voisins (n°4) sont évalués ;

7) le nouveau centre $(-4, -6)$ *et ses voisins (n°5) sont évalués ;*

8) le nouveau centre $(-2, -6)$ *et ses voisins (n°6) sont évalués ;*

9) le centre est le minimum, la distance est divisée par deux (d=1) ;

10) le centre et ses nouveaux voisins (n°7) sont évalués ;

11) le minimum final est en position $(-3, -7)$.

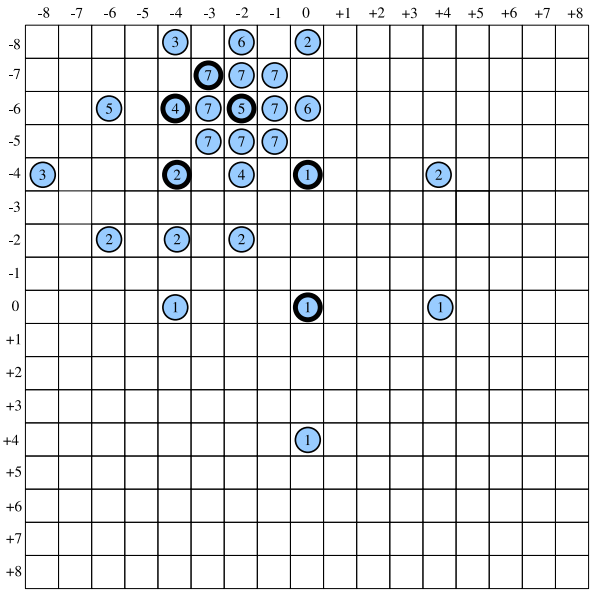


Figure B.3. *Algorithme de* bloc-matching *logarithmique*

```
[x, y]=RECHERCHE_LOGARITHMIQUE(B, Z_ref, y_B, x_B, p)
 1  // (x_B, y_B) sont les coordonnées du centre du bloc B dans la P-image
 2  // p est la distance de recherche dans l'image de référence Z_ref
 3  BI_x ← x_B − ⌊m/2⌋ − p_x
 4  BS_x ← x_B + ⌊m/2⌋ + p_x
 5  BI_y ← y_B − ⌊n/2⌋ − p_y
 6  BS_y ← y_B + ⌊n/2⌋ + p_y
 7  d_x ← ⌊p/2 + m/2⌋
 8  d_y ← ⌊p/2 + n/2⌋
 9  V_min ← MAD(B, Z_ref, y_B, x_B)
10  x_min ← x_B; y_min ← y_B
11  tant que d > 1
12  faire // trouver, dans la zone de recherche, des voisins non encore testés
13        x ← x_min
14        y ← y_min
15        pour k = 1 à 4
16        faire parmi
17              cas k = 1 : x' ← x + d_x; y' ← y;
18              cas k = 2 : x' ← x; y' ← y + d_y;
19              cas k = 3 : x' ← x − d_x; y' ← y;
20              cas k = 4 : x' ← x; y' ← y − d_y;
21            si BI_x ⩽ x' ⩽ BS_x AND BI_y ⩽ y' ⩽ BS_y
22              alors V' ← MAD(B, Zref, y', x')
23                    si V' < V_min
24                      alors V_min ← V'
25                            x_min ← x'; y_min ← y'
26        si x_min = x AND y_min = y
27          alors d_x ← ⌊d_x/2⌋ ; d_y ← ⌊d_y/2⌋
28        pour k = 1 à 8
29  faire parmi
30          cas k = 1 : x' ← x + d_x; y' ← y;
31          cas k = 2 : x' ← x + d_x; y' ← y + d_y;
32          cas k = 3 : x' ← x; y' ← y + d_y;
33          cas k = 4 : x' ← x − d_x; y' ← y + d_y;
34          cas k = 5 : x' ← x − d_x; y' ← y;
35          cas k = 6 : x' ← x − d_x; y' ← y − d_y;
36          cas k = 7 : x' ← x; y' ← y − d_y;
37          cas k = 8 : x' ← x + d_x; y' ← y − d_y;
38        si BI_x ⩽ x' ⩽ BS_x AND BI_y ⩽ y' ⩽ BS_y
39          alors V' ← MAD(B, Zref, y', x')
40                si V' < V_min
41                  alors x ← x'; y ← y'
```

Annexe C

Compléments : MPEG2

C.1. Contrôle de redondance de cycles

Lors de la transmission d'un message entre un émetteur et un récepteur, le contrôle de redondance de cycles (CRC[1]) permet de vérifier la validité du message à la réception. Le principe repose sur la division polynômiale du message envoyé. Le polynôme diviseur du polynôme est connu de l'émetteur et du récepteur.

La division étant entière (avec un reste et un quotient), l'émetteur ajoute le reste de la division au message à envoyer. Lorsque le récepteur reçoit le message codé (message + reste), celui-ci opère également la division. Si aucune erreur ne s'est introduite lors de la transmission, le reste de la division est alors nul. Si le reste n'est pas nul, il y a une (ou plusieurs) erreurs dans le message. Dans ce cas, le récepteur peut signaler l'invalidité du message pour que l'émetteur le réenvoie ou bien l'ignorer suivant le protocole établi entre eux.

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Tableau C.1. *Table de vérité de l'opérateur binaire OU exclusif*

1. *Cyclic Redundancy Check*.

EXEMPLE C.1.– *Soit le message binaire* `M = 11010110110000` *et le polynôme diviseur* $x^4 + x + 1$ *de code binaire* `10011`*. La division polynômiale se traduit, en binaire, par une séquence de ou exclusif (voir tableau C.1) :*

1) l'émetteur effectue la division entre le début du message `11010` *et le diviseur* `10011` *; le quotient prendra la valeur du bit de poids fort du dividende :* `1` *; le diviseur est multiplié par cette valeur :* `1*10011 = 10011` *; le reste est obtenu par un ou exclusif :* `11010` $\oplus$ `10011 = 01001;`

2) le prochain bit du message est ajouté au reste : `10011` *(le premier bit du reste est ignoré) ; le quotient vaut* `11` *; la division est à nouveau effectuée :* `10011` $\oplus$ `(1*10011) = 00000;`

3) le dividende vaut `00001` *; le quotient vaut :* `110` *; la division est à nouveau effectuée :* `00001` $\oplus$ `(0*10011) = 00001;`

4) le dividende vaut `00010` *; le quotient vaut :* `1100` *; la division est à nouveau effectuée :* `00010` $\oplus$ `(0*10011) = 00010;`

5) le dividende vaut `00101` *; le quotient vaut :* `11000` *; la division est à nouveau effectuée :* `00101` $\oplus$ `(0*10011) = 00101;`

6) le dividende vaut `01011` *; le quotient vaut :* `110000` *; la division est à nouveau effectuée :* `01011` $\oplus$ `(0*10011) = 01011;`

7) le dividende vaut `10110` *; le quotient vaut :* `1100001` *; la division est à nouveau effectuée :* `10110` $\oplus$ `(1*10011) = 00101;`

8) le dividende vaut `01010` *; le quotient vaut :* `11000010` *; la division est à nouveau effectuée :* `01010` $\oplus$ `(0*10011) = 01010;`

9) le dividende vaut `10100` *; le quotient vaut :* `110000101` *; la division est à nouveau effectuée :* `10100` $\oplus$ `(0*10011) = 00111;`

10) le dividende vaut `01110` *; le quotient vaut :* `1100001010` *; la division est à nouveau effectuée :* `01110` $\oplus$ `(0*10011) = 01110.`

Le reste de la division est ajouté au message :

`M' = 11010110110000+1110 = 110101101100001110.`

Le récepteur effectue la division de `M'` *par le polynôme. Si le message n'est pas altéré, le reste est nul. Si le message subit des modifications lors de la transmission, le reste ne sera pas nul.*

Annexe D

Compléments : MPEG4

D.1. L'estimation de mouvement

Afin d'affiner la précision dans l'estimation des vecteurs de déplacement du module de compensation de mouvement, le format MPEG4 propose deux niveaux de résolutions supérieurs : une résolution au demi-pixel près et une autre au quart de pixel près.

D.1.1. *Estimation de mouvement au demi-pixel*

Le principe repose sur une interpolation des valeurs des pixels. Ces interpolations prennent leurs valeurs entre les pixels. L'estimation peut alors se faire jusqu'au demi-pixel de précision. L'interpolation est volontairement simple. Suivant la figure D.1, les valeurs situées en A, B, C et D sont celles des pixels. L'interpolation pour les positions interpixelaires a, b et c valent :

$$a = \left\lfloor \frac{A+B}{2} + \frac{1}{2} \right\rfloor \tag{D.1}$$

$$b = \left\lfloor \frac{A+C}{2} + \frac{1}{2} \right\rfloor \tag{D.2}$$

$$c = \left\lfloor \frac{A+B+C+D}{4} + \frac{1}{2} \right\rfloor \tag{D.3}$$

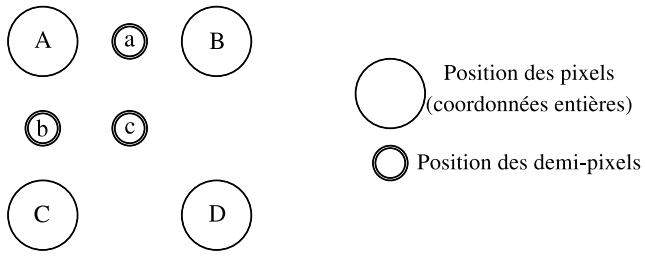


Figure D.1. *Interpolation des pixels pour une estimation à l'ordre du demi-pixel*

D.1.2. *Estimation de mouvement au quart de pixel*

Il est tentant de vouloir améliorer la précision des vecteurs de déplacement en poussant la précision jusqu'au quart de pixel.

Mais, si l'on utilise les équations précédentes (D.1, D.2 et D.3), l'erreur de prédiction (la différence entre les macroblocs de référence et le macrobloc à coder) ne diminue pas suffisamment pour contrebalancer le surcoût engendré au niveau du débit binaire.

Pour être efficace, l'interpolation au quart de pixel procède en deux étapes. La première étape calcule les valeurs des demi-pixels avec une plus grande précision que lors de l'estimation au demi-pixel.

Le voisinage est étendu au huit voisins directs du demi-pixel à interpoler. Si le demi-pixel est situé sur une ligne, le voisinage est constitué de quatre pixels à gauche du demi-pixel et de quatre pixels à droite. Si le pixel appartient à une colonne, le voisinage correspond au quatre pixels supérieurs et au quatre pixels inférieurs. La moyenne pondérée de ce voisinage définit la valeur du demi-pixel :

$$u_i = \sum_{j=1}^{8} \alpha_j \cdot A_{j,i} \text{ et } v_i = \sum_{j=1}^{8} \alpha_j \cdot A_{i,j}$$

avec :

$$(\alpha_j)_j = [-8, +24, -48, +160, +160, -48, +24, -8]/256$$

Ensuite le procédé est répété avec les demi-pixels nouvellement calculés. On obtient les valeurs des demi-pixels situés entre deux lignes et deux colonnes (voir figure D.2) :

$$w = \frac{1}{2}\left(\sum_{j=1}^{8} \alpha_j \cdot u_j + \sum_{j=1}^{8} \alpha_j \cdot v_j\right)$$

Puis la deuxième étape calcule les valeurs des quarts de pixels. Le principe est identique à celui de l'estimation de mouvement au demi-pixel (voir equations (D.1), (D.2) et (D.3) et figure D.1) mise à part que les pixels sont remplacés par les demi-pixels [PER 02].

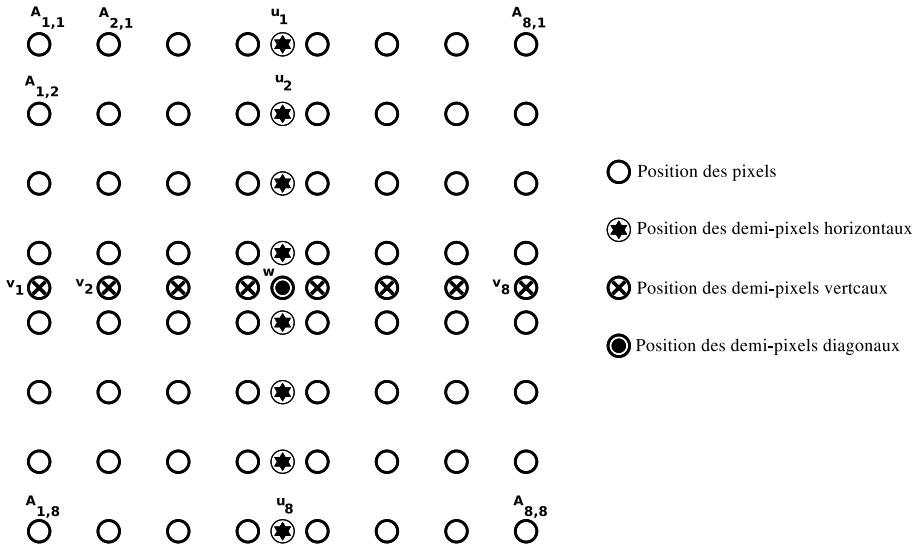


Figure D.2. *Positions des demi-pixels et des pixels de voisinage*

D.2. L'algorithme EZW

EZW est l'acronyme pour *Embedded Zerotree Wavelet*, de traduction libre : ondelettes à arborescences nulles. Cette terminologie signifie que le codage des coefficients d'une transformée en ondelettes respecte la hiérarchie qui existe au sein de ces coefficients.

L'ensemble des coefficients est structuré en une forêt dont les racines sont les coefficients des sous-bandes de la basse résolution et les feuilles les coefficients des sous-bandes de la haute résolution.

Certains arbres ou sous-arbres pouvant avoir des valeurs nulles, il sont alors indiqués par une marque spécifique (Z : *Zerotree*) afin que le codeur ou le décodeur ne les parcourt pas.

L'algorithme EZW itère trois étapes successives [SHA 93] : *étiquetage*, *affinage* et *marquage*. A chaque itération, le seuil, τ_n, est divisé par deux. L'itération s'arrête quand un seuil minimal, $\tau_{\min}$, préétabli est atteint.

```
EZW()
1  τ_1 ← valeur initiale
2  répéter
3          Liste_significatifs ← ETIQUETAGE( image )
4          AFFINAGE( Liste_significatifs )
5          MARQUAGE( Liste_significatifs )
6          τ_{n+1} ← τ_n/2
7    jusqu'à (τ_n < τ_min)
```

D.2.1. *Etiquetage*

L'*étiquetage* code les coefficients à l'aide de quatre symboles :

– P : le coefficient est supérieur au seuil τ_n. Il est significatif (c'est-à-dire supérieur en valeur absolue au seuil τ_n). Le symbole *P*ositif est codé ;

– N : le coefficient est inférieur au seuil (négatif) $-\tau_n$. Il est significatif. Le symbole *N*égatif est codé ;

– I : le coefficient est compris entre $-\tau_n$ et τ_n. Il est insignifiant. Mais certains de ses descendants sont significatifs. Le symbole *I* (valeur nulle isolée) est codé ;

– Z : le coefficient et l'ensemble de ses descendants sont insignifiants. Le symbole *Z* (arbre à valeurs nulles) est codé. Aucun descendant ne sera codé par la suite.

D.2.2. *Affinage*

L'*affinage* ajoute un bit b, au codage de chaque coefficient significatif. Ce bit dépend de l'intervalle où se situe la valeur absolue du coefficient. Initialement, il y a un seul intervalle :

$$\mathcal{I} = \{\, [\tau_1, 2\tau_1[\,\}$$

A la deuxième itération ($\tau_2 = \tau_1/2$), l'intervalle initial est scindé en deux et un nouvel intervalle est ajouté :

$$\mathcal{I} = \{\, [\tau_2, 2\tau_2[\,; [2\tau_2, 3\tau_2[\,; [3\tau_2, 4\tau_2[\,\}$$

De cette manière, les intervalles ont tous la même taille.

A la nième itération, il y a $(2^n - 1)$ intervalles :

$$\mathcal{I} = \{\, I_k = [k\tau_n, (k+1)\tau_n[\,\} \text{ avec } 1 \leqslant k < 2^n$$

Si la valeur absolue du coefficient significatif appartient à l'intervalle I_k, le *bit d'affinage* b, est mis à zéro quand la valeur absolue du coefficient est dans le demi-intervalle inférieur :

$$I_{k,0} = \left[k\tau_n, \left\lfloor \frac{k+1}{2} \right\rfloor \tau_n \right[$$

Il est mis à un quand la valeur absolue se situe dans le demi-intervalle supérieur :

$$I_{k,1} = \left[\left\lfloor \frac{k+1}{2} \right\rfloor \tau_n, (k+1)\tau_n \right]$$

D.2.3. *Marquage*

Le *marquage* des coefficients significatifs évite qu'ils soient traités par l'étape d'étiquetage des itérations suivantes.

D.2.4. *Exemple*

L'exemple provient de l'article de Jerome M. Shapiro [SHA 93].

Soit un bloc de taille 8×8 analysé par une transformée en ondelettes à trois niveaux de résolution. La figure D.3 montre le résultat de cette AMR. La parenté des coefficients est facile à observer :

– le coefficient 63 est la racine de l'arborescence ;

– le coefficient (en gras sur la figure D.3) -34, de la sous-bande verticale basse résolution, a pour descendants directs les quatre coefficients de la sous-bande verticale de résolution intermédiaire :

$$\{49, 10, 14, -13\}$$

Ses petits-enfants sont les coefficients de la sous-bande verticale de résolution maximale :

$$\{7, 13, -12, 7, 3, 4, 6, -1, 5, -7, 3, 9, 4, -2, 3, 2\}$$

– il en va de même pour les coefficients des sous-bandes horizontales et diagonales.

La figure D.4 montre l'arbre de parenté obtenu. Lors du codage (et du décodage), la première étape étiquette les coefficients. Ceux-ci sont parcourus suivant l'ordre indiqué par la figure D.5.

Le seuil τ_1, est fixé à la moitié de la plus forte amplitude :

$$\tau_1 = \left\lceil \frac{63}{2} \right\rceil = 32$$

63	**-34**	49	10	7	13	-12	7
-31	23	14	-13	3	4	6	-1
15	14	3	-12	5	-7	3	9
-9	-7	-14	8	4	-2	3	2
-5	9	-1	47	4	6	-2	2
3	0	-3	2	3	-2	0	4
2	-3	6	-4	3	6	3	6
5	11	5	6	0	3	-4	4

Figure D.3. *Résultat de l'AMR à trois niveaux sur un bloc de taille* 8×8

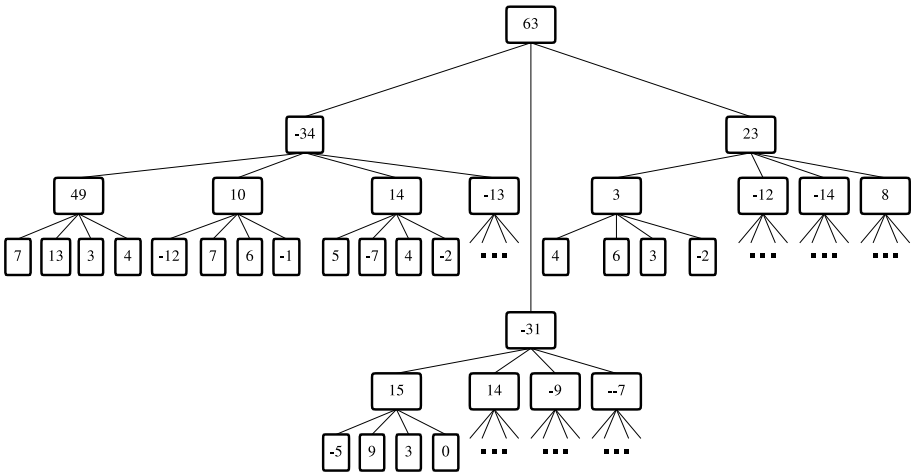


Figure D.4. *L'arbre de parenté des coefficients*

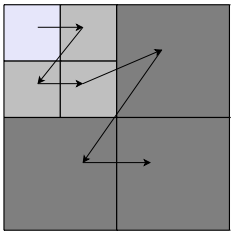


Figure D.5. *Le sens de parcours des coefficients*

L'étiquetage est le suivant :

1) le coefficient 63 étant supérieur au seuil, il est codé P. A la reconstruction, le symbole P est traduit comme étant la valeur moyenne 48 de l'intervalle positif $[32, 64[$.

2) le coefficient -34 étant supérieur au seuil, en valeur absolue, il est codé N.

3) le coefficient -31 est insignifiant puisque sa valeur absolue est inférieure au seuil. Cependant, un de ces descendants (le coefficient 47, encerclé dans la figure D.3) étant supérieur au seuil, il est codé I.

4) le coefficient 23 est également insignifiant. De plus, tous ses descendants sont également insignifiants. Ce coefficient est codé Z. Aucun descendant n'est codé par la suite.

Le tableau qui suit fournit le résultat de l'étiquetage :

Coefficient à coder	Symbole	Coefficient reconstruit
63	P	48
-34	N	-48
-31	I	0
23	Z	0
49	P	48
10	Z	0
14	Z	0
-13	Z	
15	Z	0
14	I	0
-9	Z	0
-7	Z	0
7	Z	0
13	Z	0
3	Z	0
4	Z	0
-1	Z	0
47	P	48
-3	Z	0
-2	Z	0

La deuxième étape affine le codage des coefficients significatifs. Les signes des coefficients significatifs étant codés lors de l'étiquetage, seules leurs valeurs absolues sont prises en compte par l'affinage. Chaque coefficient significatif est rangé dans le demi-intervalle qui lui correspond :

$$I_{1,0} = [32, 48[\text{ ou } I_{1,1} = [48, 64[$$

Un bit est ajouté au codage du coefficient pour indiquer le demi-intervalle auquel il appartient. La valeur moyenne 40 (resp. 56) du demi-intervalle inférieur (resp. supérieur) devient la valeur de reconstruction du coefficient. Le tableau suivant montre le résultat de l'affinage des coefficients significatifs :

Valeur absolue	63	34	49	47
Bit d'affinage	1	0	1	0
Valeur reconstruite	56	40	56	40

Le procédé est répété avec $\tau_2 = \tau_1/2 = 16$ et en ignorant les coefficients déjà significatifs :

Etiquetage : N P Z Z Z Z Z Z Z Z Z Z

Il y a, maintenant, trois intervalles :

$$\mathcal{I} = \{[16, 32[; [32, 48[; [48, 64[\}$$

Les bits d'affinage et les valeurs reconstruites des coefficients significatifs sont :

Valeur absolue	63	34	49	47	31	23
Bit d'affinage	1	0	0	1	1	1
Valeur reconstruite	60	36	52	44	28	20

Pour une valeur absolue x_i appartenant à l'intervalle I_k, le bit d'affinage est fixée suivant le demi-intervalle, $I_{k,0}$ ou $I_{k,1}$, auquel x_i appartient.

Bibliographie

[ACH 04] ACHARYA T., TSAI P.S., *JPEG2000 Standard for Image Compression : Concepts, Algorithms and VLSI Architectures*, Wiley-Interscience, Hoboken (New Jersey), 2004.

[ACH 06] ACHARYA T., CHAKRABARTI C., « A Survey on Lifting-based Discrete Wavelet Transform Architectures », *J. VLSI Signal Process. Syst.*, vol 42, n°3, p. 321-339, 2006. http://enws155.eas.asu.edu:8001/jourpapers/lifting_vlsi.pdf.

[BAR 02] BARLAUD M., LABIT C. (sous la direction de), *Compression et codage des images et des vidéos*, Hermès, Paris, 2002.

[BOU 97] BOUTELL T., PNG (Portable Network Graphics) Specification Version 1.0, Tech. Report, 1997.

[BRA 99] BRANDENBURG K., « MP3 and AAC explained », *Proc. of* $17th$ *Int'l Conf. on High Quality Audio Coding*, 1999.

[CAL 67] CALOT G., *Cours de calcul des probabilités*, Dunod, Paris, 1967.

[CIE 86] CIE N°15.2, Colorimetry, Comité internnal de l'éclairement, 1986.

[COM 87] COMPUTERSERVE INCORPORATED, GIF Graphics Interchange Format : a Standard Defining a Mechanism for the Storage and Transmission of Bitmap-Based Graphics, Tech. Report, 1987.

[COM 89] COMPUTERSERVE INCORPORATED, Graphics Interchange Format Version 89a, Tech. Report, 1990.

[COT 02] COTTET F., *Traitement des signaux et acquisition de données*, Dunod, Paris, 2002.

[DEU 96] DEUTSCH P., DEFLATE Compressed Data Format Specification Version 1.3, Tech. Report, 1996.

[GAS 00] GASQUET C., WITOMSKI P., *Analyse de Fourier et applications. Filtrage, calcul numérique et ondelettes*, Dunod, Paris, 2000.

[GER 06] GEVERS T., VAN DE WEIJER J., STOKMAN H., *Color Feature Detection*, dans R. Lukac, K.N. Plataniotis (dir.), CRC Press, Boca Raton, 2006.

[GOU 07] GOUYET J.N., MAHIEU F., « MPEG-4 : Advanced Video Coding. Systèmes et Applications », *Techniques de l'ingénieur*, TE n°5 367, 2007.

[GUI 96] GUILLOIS J.P., *Techniques de compression des images*, Hermès, Paris, 1996.

[HUB 94] HUBEL D., *L'oeil, le cerveau et la vision: les étapes cérébrales du traitement visuel*, Pour la science, Paris, 1994.

[ISO 00] ISO/IEC JTC1/SC29 WG1, JPEG 2000 Part I Final Committee Draft Version 1.0, 2000.

[LEE 96] LEE T.S., « Image Representation Using 2D Gabor Wavelets », *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 18, n° 10, p.959–971, 1996.

[MAL 89] MALLAT S., « A Theory for Multiresolution Signal Decomposition : the Wavelet Representation », *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 11, p. 674–693, 1989.

[MAL 98] MALLAT S., *Une exploration des signaux en ondelettes*, Chapitres 1 à 4, Les éditions de l'école polytechnique, Palaiseau, 1998.

[PEN 93] W. B. Pennebaker and J. L. Mitchell. *JPEG : Still Image Data Compression Standard*. Kluwer Academic Publisher, 1993.

[MAN 03] MANDAL M.K., *Multimedia signals and systems*, Kluwer Academic Publishers, Massachusetts, 2003.

[PAN 93] PAN D., « Digital Audio Compression », *Digital Tech. J.*, vol. 5, n°2, p. 28-40, 1993.

[PAN 94] PAN D., « An Overview of the MPEG/Audio Compression Algorithm », *Proc. of SPIE*, vol. 2187, pages 260-273, 1994.

[PAN 95] PAN D., « A Tutorial on MPEG/Audio Compression », *IEEE MultiMedia*, vol. 2, n°2, p. 60-74, 1995.

[PEN 93] PENNEBAKER W.B., MITCHELL J.L., *JPEG : Still Image Data Compression Standard*, Kluwer Academic Publisher, Massachusetts, 1993.

[PER 02] PEREIRA F.C., EBRAHIMI T., *The MPEG-4 Book*, Prentice Hall, VILLE?, 2002.

[PES 01] PESQUET-POPESCU B., PESQUET J.C. « Ondelettes et applications », *Techniques de l'ingénieur*, 2001.

[PNG 03] W3C AND ISO/IEC, Portable Network Graphics (PNG) Specification (Second Edition), 2003. Tech. Report N°15948.

[PRO 06] PROAKIS J.G., MANOLAKIS D.K., *Digital Signal Processing*, Chapitres 4 et 7, Prentice-Hall, New Jersey, 2006.

[PLU 94] PLUMÉ P., *Compression de données : méthodes, algorithmes, programmes détaillés*, Eyrolles, Paris, 1994.

[RIC 03] RICHARDSON I.E.G., *H264 and MPEG4 video compression : Video Coding for Next Generation Multimedia*, Willey, Chichester, 2003.

[SAL 07] SALOMON D., *Data Compression: The Complete Reference*, Springer-Verlag, London, 2007.

[SHA 93] SHAPIRO J.M., « Embedded Image Coding Using Zerotrees of Wavelet Coefficients », *IEEE Transaction on Signal Processing*, vol. 41, n° 12, p. 3445-3462, 1993.

[TAU 00] TAUBMAN D.S., « High Performance Scalable Image Compression with EBCOT », *IEEE Transactions on Image Processing*, vol 9, n°7, p. 1158–1170, 2000.

[TAU 02] TAUBMAN D.S., MARCELLIN M.W., *JPEG2000 : Image Compression Fundamentals, Standards, and Practice*, Kluwer Academic Publishers, Massachusetts, 2002.

[VAN 03] VAN VEN ENDEN A.W.M. VERHŒCKX N.A.M., *Traitement Numérique du Signal : Une introduction*, Dunod, Paris, 2003.

[WEI 96] WEINBERGER M.J., SEVUSSI G., SAPIO G., « LOCO-I : A Low Complexity, Context-Based, Lossless Image Compression Algorithm », *Proc. of Data Compression Conference*, p. 140-149, 1996, http://cs.haifa.ac.il/courses/src_coding/LOCO1_DCC1996.pdf.

[WEI 99] WEINBERGER M.J., SEROUSSI G., SAPIRO G., « From LOCO-I to the JPEG-LS standard », *Proc. of the 1999 Int'l Conference on Image Processing*, p. 68-72, 1999, http://fig.il/courses/src_coding/FromLocoToJPEG-ICIP-1999.pdf.

Notations

$a \gtrsim b$	a est proche de b tout en restant supérieure.
$< a, b >$	Produit scalaire entre a et b (voir annexe **??**).
$\xrightarrow{s} \boxed{\downarrow 2} \rightarrow$	Ce processus ne conserve qu'un échantillon sur deux de la source s.
$\xrightarrow{s} \boxed{\uparrow 2} \rightarrow$	Ce processus insère une valeur nulle après chaque échantillon de la source s.
$\hat{x}$	Valeur quantifiée proche de la valeur originale x sans y être identique.
$\tilde{x}$	Valeur prédite pour la valeur x de la source.
$\widehat{\tilde{x}}$	Valeur prédite quantifiée correspondant à la valeur x de la source.
$\mathrm{sinc}(x) = \frac{\sin(x)}{x}$	Sinus cardinal de x.
$S(f)$	Série de Fourier du signal $s(t)$.
$\hat{S}(f)$	Transformée de Fourier du signal $s(t)$.
$x(t) \otimes g(t)$	Convolution du signal $x(t)$ par le filtre $g(t)$.
P^T	Transposée de la matrice P. Les lignes de P sont les colonnes de P^T : $P^T = (p_{j,i})_{i,j}$ si $P = (p_{i,j})_{i,j}$
v^T	Vecteur ligne, transposé du vecteur (colonne)v. Par exemple : $v = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ et $v^T = (a, b, c)$ Remarque : soit deux vecteurs u et v, le produit scalaire $< u, v >= u^T \cdot v$
$\lfloor x \rfloor$	Partie entière du réel x.
$\lceil x \rceil$	$\lfloor x \rfloor + 1$.
$\mathrm{round}(x)$	Arrondie du réel x à l'entier le plus proche avec la valeur 0,5 ramenée à 0.
$\lvert x \rvert$	Valeur absolue de x.

$x \bmod y$	Reste de la division entière de x par y.
$\text{sign}(x) = \begin{cases} +1 & \textbf{si } x > 0 \\ -1 & \textbf{si } x < 0 \\ 0 & \textbf{sinon} \end{cases}$	Signe de x.
$e^{i\theta} = \cos(\theta) + i\sin(\theta)$	Exponentielle complexe. $\Longrightarrow \begin{cases} \cos(\theta) &= \frac{1}{2}\left(e^{i\theta} + e^{-i\theta}\right) \\ \sin(\theta) &= -\frac{i}{2}\left(e^{i\theta} - e^{-i\theta}\right) \end{cases}$
$\log_k(x) = \frac{\ln(x)}{\ln(k)}$	Logarithme en base k de x.
$\ln(x)$	Logarithme naturel de x
$\underset{l}{\text{argmin}}(f(x,l))$	Retourne la valeur de l t.q. $f(x,l)$ soit minimale.
$\underset{l}{\text{argmax}}(f(x,l))$	Retourne la valeur de l t.q. $f(x,l)$soit maximale.

Glossaire

4MV *Four Motion Vectors* (MPEG4) 172

ADPCM *Adaptive Differential Pulse Code Modulation* 33

AFX *Animation Framwork eXtension* 157

API *Application Portability Interface* 151

API Interface de Portabilité d'Application 151

AVC *Advanced Video Coding* 157

AVO *AudioVisual Object* : objet audiovisuel (MPEG4) 197

BAB *Binary Alpha Bloc* 180

BIFS *BInary Format for Scenes* 160

Boîtier TV Ce terme regroupe un ensemble de composants électroniques et d'algorithmes qui permettent la réception et le décodage d'émissions télédiffusées. Un ensemble de fonctionnalités, plus ou moins avancées, sont possible grâce à ce boîtier 127

CABAC *Contextual Adaptive Binary Arithmetic Coding* 208

CABAC Codage Arithmétique Binaire Adapté aux Contextes 208

CAT *Conditional Access Table* 130

CAT Table des Accès Conditionnés 130

CAVLC *Content Based Variable Length Coding* 206

CAVLC Codage à longueur variable contextuel 206

Index

Q, R

S

T, U, V, W

SOMMAIRE

Le multimédia et la compression